

Fast Covariance Estimation for Sparse Functional Data

Luo Xiao^{1,*}, Cai Li¹, Will Checkley² and Ciprian M. Crainiceanu²

¹ North Carolina State University and ²Johns Hopkins University

*email: lxiao5@ncsu.edu

January 9, 2016

Abstract

We propose a novel covariance smoothing method and associated software based on penalized spline smoothing. The proposed method is a bivariate smoother that is designed for covariance smoothing and can be used for sparse functional or longitudinal data. We propose a fast algorithm for covariance smoothing using leave-one-subject-out cross validation. Our simulations demonstrate that the proposed method compares favorably against several commonly used methods. The method is applied to a study of child growth led by one of coauthors and to a public dataset of longitudinal CD4 counts.

Keywords: *bivariate smoothing; face; functional data analysis; fPCA; penalized splines.*

1 Introduction

The covariance function is a crucial ingredient in functional data analysis. Sparse functional or longitudinal data are ubiquitous in scientific studies, while functional principal component analysis has become one of the first-line approaches to analyzing this type of data; see, e.g., Besse and Ramsay (1986);

Ramsay and Dalzell (1991); Kneip (1994); Besse et al. (1997); Staniswalis and Lee (1998); Yao et al. (2003, 2005).

Given a sample of functions observed at a finite number of locations and, often, with sizable measurement error, there are usually three approaches for obtaining smooth functional principal components: 1) smooth the functional principal components of the sample covariance function; 2) smooth each curve and diagonalize the resulting sample covariance of the smoothed curves; and 3) smooth the sample covariance function and then diagonalize it.

Smoothing of the sample covariance function is typically done using bivariate smoothing. Local linear smoothers (Fan and Gijbels, 1996), tensor-product bivariate P -splines (Eilers and Marx, 2003) and thin plate regression splines (Wood, 2003) are among the popular methods for smoothing the sample covariance function. For example, the *fpca.sc* function in the R package *refund* (Huang et al., 2015) uses the tensor-product bivariate P -splines. However, there are two known problems with these smoothers: 1) they are general-purpose smoothers that are not designed specifically for covariance operators; and 2) they ignore that the subject, instead of the observation, is the independent sampling unit and assume that the empirical covariance surface is the sum between an underlying smooth covariance surface and independent random noise. The FACE smoothing approach proposed by Xiao et al. (2014) was designed specifically for addressing these weaknesses of off-the-shelf covariance smoothing software. The method is implemented in the function *fpca.face* in the *refund* R package (Huang et al., 2015) and has proven to be reliable and fast in a range of applications. However, FACE was developed for high-dimensional dense functional data and the extension to sparse data is far from obvious. One approach that attempts to solve these problems was proposed by Yao et al. (2003). In their paper they used leave-one-subject-out cross-validation to choose the bandwidth for local polynomial smoothing methods. This approach is theoretically sound, but computationally expensive. This may be the reason why the practice is to either try multiple bandwidths and visually inspect the results or completely ignore within-subject correlations.

Several alternative methods for covariance smoothing of sparse functional data also exist in the literature: James et al. (2000) used reduced rank spline mixed effects models while Peng and Paul (2009) proposed a geometric approach under the framework of marginal maximum likelihood estimation.

This paper has two aims. First, we propose a new automatic bivariate smoother that is specifically designed for covariance function estimation and can be used for sparse functional data. Second, we propose a fast algorithm for selecting the smoothing parameter using leave-one-subject-out cross validation. The R code for the proposed method will be made publicly available in the *face* package.

2 Model

Suppose that the observed data take the form $\{(y_{ij}, t_{ij}), j = 1, \dots, m_i, i = 1, \dots, n\}$, where t_{ij} is in the unit interval $[0, 1]$, n is the number of subjects, and m_i is the number of observations for subject i . The model is

$$y_{ij} = f(t_{ij}) + u_i(t_{ij}) + \epsilon_{ij}, \quad (1)$$

where f is a smooth mean function, $u_i(t)$ is generated from a zero-mean Gaussian process with covariance operator $\mathcal{C}(s, t) = \text{Cov}\{u_i(s), u_i(t)\}$, and ϵ_{ij} is white noise following a normal distribution $\mathcal{N}(0, \sigma_\epsilon^2)$. We assume that the random terms are independent across subjects and from each other. In many cases when data are sparse, m_i 's are usually much smaller than n .

We are interested in estimating the covariance function $\mathcal{C}(s, t)$ and we assume that a working estimator \hat{f} exists. In our code, we use a P -spline smoother (Eilers and Marx, 1996) with the smoothing parameter selected by leave-one-subject-out cross validation. A standard procedure employed for obtaining a smooth estimate of $\mathcal{C}(s, t)$ consists of two steps. In the first step, an empirical estimate of the covariance function is constructed: $\{\hat{C}_{ij_1j_2} : 1 \leq j_1 \neq j_2 \leq m_i, i = 1, \dots, n\}$, where $\hat{C}_{ij_1j_2} = r_{ij_1}r_{ij_2}$ and $r_{ij} = y_{ij} - \hat{f}(t_{ij})$. In the second step, the raw estimates are smoothed using a bivariate

smoother. Standard bivariate smoothers are local linear smoothers (Fan and Gijbels, 1996), tensor-product bivariate P -splines (Eilers and Marx, 2003) and thin plate regression splines (Wood, 2003). In the following section we propose a statistically efficient, computationally fast and automatic smoothing procedure that serves as competitive alternative to standard approaches.

3 Method

We model the covariance function $\mathcal{C}(s, t)$ as a tensor-product splines $\mathcal{H}(s, t) = \sum_{1 \leq \kappa \leq c, 1 \leq \ell \leq c} \theta_{\kappa\ell} B_\kappa(s) B_\ell(t)$, where $\Theta = (\theta_{\kappa\ell})_{1 \leq \kappa \leq c, 1 \leq \ell \leq c}$ is a coefficient matrix, $\{B_1(\cdot), \dots, B_c(\cdot)\}$ is the collection of B-spline basis functions in the unit interval, and c is the number of interior knots plus the order (degree plus 1) of the B-splines. We use knots placed at the quantiles of observed t and enforce the following constraint on Θ :

$$\theta_{\kappa\ell} = \theta_{\ell\kappa}, 1 \leq \kappa, \ell \leq c.$$

With this constraint, $\mathcal{H}(s, t)$ is always symmetric in s and t , a desired property for estimates of covariance functions. Unlike the standard approach for estimating smooth covariance function, our method differs in two directions: first, our approach applies a joint estimation of covariance function and error variance; second, we incorporate the correlation structure of the auxiliary variables $\{\hat{C}_{ij_1j_2} : 1 \leq j_1 \leq j_2 \leq m_i, i = 1, \dots, n\}$ in a two-step procedure to boost statistical efficiency. Because we use a relatively large number of knots, estimating Θ by least squares or weighted least squares tends to overfit. Thus, we estimate Θ by minimizing the following penalized weighted least squares criterion

$$(\hat{\Theta}, \hat{\sigma}_\epsilon^2) = \arg \min_{\Theta: \Theta = \Theta^T, \sigma_\epsilon^2} \sum_{i=1}^n \left(\mathbf{H}_i + \boldsymbol{\delta}_i \sigma_\epsilon^2 - \hat{\mathbf{C}}_i \right)^T \mathbf{W}_i \left(\mathbf{H}_i + \boldsymbol{\delta}_i \sigma_\epsilon^2 - \hat{\mathbf{C}}_i \right) + \lambda \cdot \text{tr}(\Theta \mathbf{D} \mathbf{D}^T \Theta^T), \quad (2)$$

where $n_i = m_i(m_i+1)/2$, $\hat{\mathbf{C}}_i = (\hat{\mathbf{C}}_{i1}^T, \hat{\mathbf{C}}_{i2}^T, \dots, \hat{\mathbf{C}}_{im_i}^T)^T \in \mathbb{R}^{n_i}$, $\mathbf{H}_i = (\mathbf{H}_{i1}^T, \mathbf{H}_{i2}^T, \dots, \mathbf{H}_{im_i}^T)^T \in \mathbb{R}^{n_i}$, and $\boldsymbol{\delta}_i = (\boldsymbol{\delta}_{i1}, \boldsymbol{\delta}_{i2}, \dots, \boldsymbol{\delta}_{im_i})^T \in \mathbb{R}^{n_i}$ with $\hat{\mathbf{C}}_{ij} = (\hat{C}_{ijj}, \hat{C}_{ij(j+1)}, \dots, \hat{C}_{ijm_i})^T \in \mathbb{R}^{m_i-j+1}$, $\mathbf{H}_{ij} = \{\mathcal{H}(t_{ij}, t_{ij}), \mathcal{H}(t_{ij}, t_{i(j+1)}), \dots, \mathcal{H}(t_{ij}, t_{im_i})\}^T \in \mathbb{R}^{m_i-j+1}$, and

$\boldsymbol{\delta}_{ij} = (1, \mathbf{0}_{m_i-j}^T)^T \in \mathbb{R}^{m_i-j+1}$ for $1 \leq j \leq m_i$. Also \mathbf{W}_i is a weight matrix, $\text{tr}(\cdot)$ is the sum of all diagonal entries of a square matrix, \mathbf{D} is the second-order differencing matrix (Eilers and Marx, 1996), and λ is a smoothing parameter that balances model fit and smoothness of the estimate. The penalty term $\text{tr}(\boldsymbol{\Theta} \mathbf{D} \mathbf{D}^T \boldsymbol{\Theta}^T)$ can be interpreted as the row penalty in bivariate P -splines (Eilers and Marx, 2003). Note that when $\boldsymbol{\Theta}$ is symmetric, as in our case, the row and column penalties in bivariate P -splines become the same. Therefore, our proposed method can be regarded as a special case of bivariate P -splines that is designed specifically for covariance function estimation. It follows that our estimate is given by $\tilde{C}(s, t) = \sum_{1 \leq \kappa \leq c, 1 \leq \ell \leq c} \hat{\theta}_{\kappa\ell} B_\kappa(s) B_\ell(t)$.

3.1 Estimation

Let $\mathbf{b}(t) = (B_1(t), \dots, B_c(t))^T$ be a column vector. Then $\mathcal{H}(s, t) = \{\mathbf{b}(t) \otimes \mathbf{b}(s)\}^T \text{vec} \boldsymbol{\Theta}$. Here $\text{vec}(\cdot)$ is an operator that stacks the columns of a matrix into a column vector. Let $\boldsymbol{\theta} = \text{vech} \boldsymbol{\Theta}$, where $\text{vech}(\cdot)$ is an operator that stacks the columns of the lower triangle of a matrix into a column vector, and let \mathbf{G}_c be the duplication matrix (page 246, Seber 2007) such that $\text{vec} \boldsymbol{\Theta} = \mathbf{G}_c \boldsymbol{\theta}$. It follows that $\mathcal{H}(s, t) = \{\mathbf{b}(t) \otimes \mathbf{b}(s)\}^T \mathbf{G}_c \boldsymbol{\theta}$. Next we let $\mathbf{B}_i = [\mathbf{B}_{i1}^T, \dots, \mathbf{B}_{im_i}^T]^T$, where $\mathbf{B}_{ij} = [\mathbf{b}(t_{ij}), \dots, \mathbf{b}(t_{im_i})] \otimes \mathbf{b}(t_{ij})$. Then, we have

$$\begin{aligned} \mathbb{E}(\hat{\mathbf{C}}_i) &= \mathbf{H}_i + \boldsymbol{\delta}_i \sigma_\epsilon^2 \\ &= \begin{pmatrix} \mathbf{B}_i \mathbf{G}_c & \boldsymbol{\delta}_i \end{pmatrix} \begin{pmatrix} \boldsymbol{\theta} \\ \sigma_\epsilon^2 \end{pmatrix} \\ &= \mathbf{X}_i \boldsymbol{\alpha}, \end{aligned}$$

and

$$\begin{aligned} \sum_{i=1}^n \left(\mathbf{H}_i + \boldsymbol{\delta}_i \sigma_\epsilon^2 - \hat{\mathbf{C}}_i \right)^T \mathbf{W}_i \left(\mathbf{H}_i + \boldsymbol{\delta}_i \sigma_\epsilon^2 - \hat{\mathbf{C}}_i \right) \\ = (\hat{\mathbf{C}} - \mathbf{X} \boldsymbol{\alpha})^T \mathbf{W} (\hat{\mathbf{C}} - \mathbf{X} \boldsymbol{\alpha}), \end{aligned} \quad (3)$$

where $\hat{\mathbf{C}} = (\hat{\mathbf{C}}_1^T, \dots, \hat{\mathbf{C}}_n^T)^T$, $\mathbf{B} = [\mathbf{B}_1^T, \dots, \mathbf{B}_n^T]^T$, $\boldsymbol{\delta} = (\boldsymbol{\delta}_1^T, \dots, \boldsymbol{\delta}_n^T)^T$, $\mathbf{X} = \begin{pmatrix} \mathbf{B} \mathbf{G}_c & \boldsymbol{\delta} \end{pmatrix}$, and $\mathbf{W} = \text{blockdiag}(\mathbf{W}_1, \dots, \mathbf{W}_n)$. Next we can derive that (page

241, Seber 2007)

$$\text{tr}(\mathbf{\Theta} \mathbf{D} \mathbf{D}^T \mathbf{\Theta}^T) = (\text{vec } \mathbf{\Theta})^T (\mathbf{I}_c \otimes \mathbf{D} \mathbf{D}^T) \text{vec } \mathbf{\Theta}.$$

Because $\text{vec } \mathbf{\Theta} = \mathbf{G}_c \boldsymbol{\theta}$, we obtain that

$$\begin{aligned} \text{tr}(\mathbf{\Theta} \mathbf{D} \mathbf{D}^T \mathbf{\Theta}^T) &= \boldsymbol{\theta}^T \mathbf{G}_c^T (\mathbf{I}_c \otimes \mathbf{D} \mathbf{D}^T) \mathbf{G}_c^T \boldsymbol{\theta} \\ &= \begin{pmatrix} \boldsymbol{\theta}^T & \sigma_\epsilon^2 \end{pmatrix} \begin{pmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\theta} \\ \sigma_\epsilon^2 \end{pmatrix} \\ &= \boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha}, \end{aligned} \quad (4)$$

where $\mathbf{P} = \mathbf{G}_c^T (\mathbf{I}_c \otimes \mathbf{D} \mathbf{D}^T) \mathbf{G}_c^T$, \mathbf{Q} is the block matrix containing \mathbf{P} and zeros.

By (3) and (4), the objective function in (2) can be rewritten as

$$\hat{\boldsymbol{\alpha}} = \arg \min_{\boldsymbol{\alpha}} (\hat{\mathbf{C}} - \mathbf{X} \boldsymbol{\alpha})^T \mathbf{W} (\hat{\mathbf{C}} - \mathbf{X} \boldsymbol{\alpha}) + \lambda \boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha}. \quad (5)$$

Now we obtain an explicit form of $\hat{\boldsymbol{\alpha}}$

$$\hat{\boldsymbol{\alpha}} = \begin{pmatrix} \hat{\boldsymbol{\theta}} \\ \hat{\sigma}_\epsilon^2 \end{pmatrix} = (\mathbf{X}^T \mathbf{W} \mathbf{X} + \lambda \mathbf{Q})^{-1} (\mathbf{X}^T \mathbf{W} \hat{\mathbf{C}}).$$

We need to specify the weight matrices \mathbf{W}_i 's. One sensible choice for \mathbf{W}_i is the inverse of $\text{Cov}(\hat{\mathbf{C}}_i)$. However $\text{Cov}(\hat{\mathbf{C}}_i)$ may not be invertible. Thus, we specify \mathbf{W}_i as

$$\mathbf{W}_i^{-1} = (1 - \beta) \text{Cov}(\hat{\mathbf{C}}_i) + \beta \text{diag}\{\text{diag}\{\text{Cov}(\hat{\mathbf{C}}_i)\}\}, 1 \leq i \leq n,$$

for some constant $0 < \beta < 1$. The above specification ensures that \mathbf{W}_i^{-1} exists. We will use $\beta = 0.05$, which works well in practice.

We now derive $\text{Cov}(\hat{\mathbf{C}}_i)$ in terms of \mathcal{C} and σ_ϵ^2 . First note that

$$\mathbb{E}(r_{ij_1} r_{ij_2}) = \text{Cov}(r_{ij_1}, r_{ij_2}) = \mathcal{C}(t_{ij_1}, t_{ij_2}) + \delta_{j_1 j_2} \sigma_\epsilon^2.$$

Proposition 1. Let $\mathbf{M}_1 = (\mathcal{C}(t_{ij_1}, t_{ij_3}), \delta_{j_1 j_3} \sigma_\epsilon^2)^T$, $\mathbf{M}_2 = (\mathcal{C}(t_{ij_1}, t_{ij_4}), \delta_{j_1 j_4} \sigma_\epsilon^2)^T$; $\mathbf{M}'_1 = (\mathcal{C}(t_{ij_2}, t_{ij_4}), \delta_{j_2 j_4} \sigma_\epsilon^2)^T$, $\mathbf{M}'_2 = (\mathcal{C}(t_{ij_2}, t_{ij_3}), \delta_{j_2 j_3} \sigma_\epsilon^2)^T$. $\mathbf{M} = [\mathbf{M}_1 | \mathbf{M}_2]$, $\mathbf{M}' = [\mathbf{M}'_1 | \mathbf{M}'_2]$, then

$$\text{Cov}(\hat{\mathbf{C}}_{ij_1 j_2}, \hat{\mathbf{C}}_{ij_3 j_4}) = \sum_{j_1, j_2, j_3, j_4} (\text{vec}(\mathbf{M} \odot \mathbf{M}')) = \sum_{j_1, j_2, j_3, j_4} (\text{vec}([\mathbf{M}_1 \otimes \mathbf{M}'_1 | \mathbf{M}_2 \otimes \mathbf{M}'_2])),$$

where \odot is the Khatri-Rao product.

The proof of Proposition 1 is provided in the appendix. Now we see that \mathbf{W}_i also depends on $(\mathcal{C}, \sigma_\epsilon^2)$. Hence, we employ a two-step estimation. We first estimate $(\mathcal{C}, \sigma_\epsilon^2)$ by fitting ordinary least square (OLS), i.e., $\mathbf{W}_i = \mathbf{I}$ for all i , then we obtain the plug-in estimate of \mathbf{W}_i and estimate $(\mathcal{C}, \sigma_\epsilon^2)$ using generalized least square (GLS). The algorithm for the two-step estimation is summarized as Algorithm 1.

Algorithm 1 Estimation algorithm

$$\hat{\boldsymbol{\alpha}}^{(0)} = \arg \min_{\boldsymbol{\alpha}} (\hat{\mathbf{C}} - \mathbf{X}\boldsymbol{\alpha})^T (\hat{\mathbf{C}} - \mathbf{X}\boldsymbol{\alpha}) + \lambda \boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha}$$

$$\widehat{\mathbf{W}} = \mathbf{W}(\hat{\boldsymbol{\alpha}}^{(0)})$$

$$\hat{\boldsymbol{\alpha}} = \arg \min_{\boldsymbol{\alpha}} (\hat{\mathbf{C}} - \mathbf{X}\boldsymbol{\alpha})^T \widehat{\mathbf{W}} (\hat{\mathbf{C}} - \mathbf{X}\boldsymbol{\alpha}) + \lambda \boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha}$$

3.2 Selection of the smoothing parameter

For selecting the smoothing parameter, we use leave-one-subject-out cross validation, a popular approach for correlated data; see, for example, Yao et al. (2003), Reiss et al. (2010) and Xiao et al. (2015). Compared to the leave-one-observation-out cross validation, which ignores the correlation, leave-one-subject-out cross-validation was reported to be more robust against overfit. However, such an approach is usually computationally expensive. In this section, we derive a fast algorithm for approximating the leave-one-subject-out cross validation.

Let $\tilde{\mathbf{C}}_i^{[i]}$ be the prediction of $\hat{\mathbf{C}}_i$ by applying the proposed method to the data without the data from the i th subject, then the cross-validated error is

$$\text{iCV} = \sum_{i=1}^n \|\tilde{\mathbf{C}}_i^{[i]} - \hat{\mathbf{C}}_i\|^2. \quad (6)$$

There is a simple formula for iCV. First we let $\mathbf{S} = \mathbf{X}(\mathbf{X}^T \mathbf{W} \mathbf{X} + \lambda \mathbf{Q})^{-1} \mathbf{X}^T \mathbf{W}$, which is the smoother matrix for the proposed method. \mathbf{S} can be written as $(\mathbf{X}\mathbf{A})[\mathbf{I} + \lambda \text{diag}(\mathbf{s})]^{-1} (\mathbf{X}\mathbf{A})^T \mathbf{W}$ for some orthogonal square matrix \mathbf{A} satisfying $\mathbf{A}^T \mathbf{A} = \mathbf{I}$ and \mathbf{s} is a column vector; see, for example, Xiao et al. (2013). In particular, both \mathbf{A} and \mathbf{s} do not depend on λ .

Let $\mathbf{S}_i = \mathbf{X}_i(\mathbf{X}^T \mathbf{W} \mathbf{X} + \lambda \mathbf{Q})^{-1} \mathbf{X}_i^T \mathbf{W}$ and $\mathbf{S}_{ii} = \mathbf{X}_i(\mathbf{X}^T \mathbf{W} \mathbf{X} + \lambda \mathbf{Q})^{-1} \mathbf{X}_i^T \mathbf{W}_i$. Then \mathbf{S}_i is of dimension $n_i \times N$, where $N = \sum_{i=1}^n n_i$, and \mathbf{S}_{ii} is symmetric and of dimension $n_i \times n_i$.

Lemma 1. *The iCV in (6) can be simplified as*

$$\text{iCV} = \sum_{i=1}^n \|(\mathbf{I}_{n_i} - \mathbf{S}_{ii})^{-1}(\mathbf{S}_i \hat{\mathbf{C}} - \hat{\mathbf{C}}_i)\|^2.$$

The proof of Lemma 1 is the same as that of Lemma 3.1 in Xu and Huang (2012) and thus is omitted. Similar to Xu and Huang (2012), we further simplify iCV by using the approximation $(\mathbf{I}_{n_i} - \mathbf{S}_{ii})^{-2} = \mathbf{I}_{n_i} + 2\mathbf{S}_{ii}$. Note that \mathbf{S}_{ii} is symmetric. This approximation leads to the generalized cross validation, which we denote as iGCV,

$$\begin{aligned} \text{iGCV} &= \sum_{i=1}^n (\mathbf{S}_i \hat{\mathbf{C}} - \hat{\mathbf{C}}_i)^T (\mathbf{I}_{n_i} + 2\mathbf{S}_{ii}) (\mathbf{S}_i \hat{\mathbf{C}} - \hat{\mathbf{C}}_i) \\ &= \|\hat{\mathbf{C}} - \mathbf{S} \hat{\mathbf{C}}\|^2 + 2 \sum_{i=1}^n (\mathbf{S}_i \hat{\mathbf{C}} - \hat{\mathbf{C}}_i)^T \mathbf{S}_{ii} (\mathbf{S}_i \hat{\mathbf{C}} - \hat{\mathbf{C}}_i). \end{aligned} \quad (7)$$

While iGCV in (7) is much easier to compute than iCV in (6), the formula in (7) is still computationally expensive as the smoother matrix \mathbf{S} is of dimension $N \times N$, where $N = 2,000$ if $n = 100$ and $m_i = m = 5$ for all i . Thus, we further simplify iGCV.

Let $\mathbf{F}_i = \mathbf{X}_i \mathbf{A}$, $\mathbf{F} = \mathbf{X} \mathbf{A}$ and $\tilde{\mathbf{F}} = \mathbf{F}^T \mathbf{W}$. Define $\mathbf{f}_i = \mathbf{F}_i^T \hat{\mathbf{C}}_i$, $\mathbf{f} = \mathbf{F}^T \hat{\mathbf{C}}$, $\tilde{\mathbf{f}} = \tilde{\mathbf{F}} \hat{\mathbf{C}}$, $\mathbf{J}_i = \mathbf{F}_i^T \mathbf{W}_i \hat{\mathbf{C}}_i$ and $\mathbf{L}_i = \mathbf{F}_i^T \mathbf{W}_i \mathbf{F}_i$. To simplify notation we will denote $[\mathbf{I} + \lambda \text{diag}(\mathbf{s})]^{-1}$ as $\tilde{\mathbf{D}}$, a symmetric matrix, and its diagonal as $\tilde{\mathbf{d}}$.

Proposition 2. *The iGCV in (7) can be simplified as*

$$\text{iGCV} = \|\hat{\mathbf{C}}\|^2 - 2\tilde{\mathbf{d}}^T (\tilde{\mathbf{f}} \odot \mathbf{f}) + (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}})^T (\mathbf{F}^T \mathbf{F}) (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}}) + 2\tilde{\mathbf{d}}^T \mathbf{g} - 4\tilde{\mathbf{d}}^T \mathbf{G}_1 \tilde{\mathbf{d}} + 2\tilde{\mathbf{d}}^T \mathbf{G}_2 \left\{ (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}}) \otimes (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}}) \right\},$$

where $\mathbf{g} = \sum_{i=1}^n (\mathbf{J}_i \odot \mathbf{f}_i)$, $\mathbf{G}_1 = \sum_{i=1}^n \{(\mathbf{J}_i \tilde{\mathbf{f}}^T) \odot (\mathbf{F}_i^T \mathbf{F}_i)\}$, and $\mathbf{G}_2 = \sum_{i=1}^n (\mathbf{L}_i \odot \mathbf{F}_i^T \mathbf{F}_i)$. Here \odot is the row-wise Khatri-Rao product.

Remark 1. *While the above formula looks complex, it can be efficiently computed. Indeed, only the term $\tilde{\mathbf{d}}$ depend on the smoothing parameter λ and it can be easily computed; all other terms including \mathbf{g} , \mathbf{G}_1 , \mathbf{G}_2 can be pre-calculated just for once.*

The proof of Proposition 2 is provided in the appendix.

4 Prediction

In this section, we consider the prediction of $X_i(t) = f(t_i) + u_i(t_{ij})$, subject i 's curve. We assume that $X_i(t)$ is generated from a Gaussian process. Suppose we would like to predict $X_i(t)$ at $\{s_{i1}, \dots, s_{im}\}$ for $m \geq 1$. Let $\mathbf{y}_i = (\mathbf{y}_{i1}, \dots, \mathbf{y}_{im_i})^T$, $\mathbf{f}_i^o = \{f(t_{i1}), \dots, f(t_{im_i})\}^T$, and $\mathbf{x}_i = \{X_i(s_{i1}), \dots, X_i(s_{im})\}^T$. Let $\mathbf{H}_i^o = \{\mathbf{b}(t_{i1})^T, \dots, \mathbf{b}(t_{im_i})^T\}^T$, $\mathbf{H}_i^n = \{\mathbf{b}(s_{i1})^T, \dots, \mathbf{b}(s_{im})^T\}^T$, and $\mathbf{H}_i = [\mathbf{H}_i^{o,T}, \mathbf{H}_i^{n,T}]^T$. Then

$$\begin{pmatrix} \mathbf{y}_i \\ \mathbf{x}_i \end{pmatrix} \sim \text{MVN} \left\{ \begin{pmatrix} \mathbf{f}_i^o \\ \mathbf{f}_i^n \end{pmatrix}, \begin{pmatrix} \mathbf{H}_i^o \boldsymbol{\Theta} \mathbf{H}_i^{o,T} + \sigma_\epsilon^2 \mathbf{I}_{m_i} & \mathbf{H}_i^o \boldsymbol{\Theta} \mathbf{H}_i^{n,T} \\ \mathbf{H}_i^n \boldsymbol{\Theta} \mathbf{H}_i^{o,T} & \mathbf{H}_i^n \boldsymbol{\Theta} \mathbf{H}_i^{n,T} + \sigma_\epsilon^2 \mathbf{I}_m \end{pmatrix} \right\}$$

We derive that

$$\mathbb{E}(\mathbf{x}_i | \mathbf{y}_i) = \left(\mathbf{H}_i^n \boldsymbol{\Theta} \mathbf{H}_i^{o,T} \right) \mathbf{V}_i^{-1} (\mathbf{y}_i - \mathbf{f}_i^o) + \mathbf{f}_i^n,$$

where $\mathbf{V}_i = \mathbf{H}_i^o \boldsymbol{\Theta} \mathbf{H}_i^{o,T} + \sigma_\epsilon^2 \mathbf{I}_{m_i}$, and

$$\text{Cov}(\mathbf{x}_i | \mathbf{y}_i) = \mathbf{V}_i^n - \left(\mathbf{H}_i^n \boldsymbol{\Theta} \mathbf{H}_i^{o,T} \right) \mathbf{V}_i^{-1} \left(\mathbf{H}_i^n \boldsymbol{\Theta} \mathbf{H}_i^{o,T} \right)^T,$$

where $\mathbf{V}_i^n = \mathbf{H}_i^n \boldsymbol{\Theta} \mathbf{H}_i^{n,T} + \sigma_\epsilon^2 \mathbf{I}_m$. Because f , $\boldsymbol{\Theta}$ and σ_ϵ^2 are unknown, we need to plug in their estimates \hat{f} , $\hat{\boldsymbol{\Theta}}$ and $\hat{\sigma}_\epsilon^2$, respectively, into the above equalities.

Thus, we could predict \mathbf{x}_i by

$$\hat{\mathbf{x}}_i = \{\hat{x}(s_{i1}), \dots, \hat{x}(s_{im})\}^T = \left(\mathbf{H}_i^n \hat{\boldsymbol{\Theta}} \mathbf{H}_i^{o,T} \right) \hat{\mathbf{V}}_i^{-1} (\mathbf{y}_i - \hat{\mathbf{f}}_i^o) + \hat{\mathbf{f}}_i^n,$$

where $\hat{\mathbf{f}}_i^o = \{\hat{f}(t_{i1}), \dots, \hat{f}(t_{im_i})\}^T$, $\hat{\mathbf{f}}_i^n = \{\hat{f}(s_{i1}), \dots, \hat{f}(s_{im})\}^T$, and $\hat{\mathbf{V}}_i = \mathbf{H}_i^o \hat{\boldsymbol{\Theta}} \mathbf{H}_i^{o,T} + \hat{\sigma}_\epsilon^2 \mathbf{I}_{m_i}$. Moreover, an approximate covariance matrix for $\hat{\mathbf{x}}_i$ is

$$\widehat{\text{Cov}}(\hat{\mathbf{x}}_i | \mathbf{y}_i) = \hat{\mathbf{V}}_i^n - \left(\mathbf{H}_i^n \hat{\boldsymbol{\Theta}} \mathbf{H}_i^{o,T} \right) \hat{\mathbf{V}}_i^{-1} \left(\mathbf{H}_i^n \hat{\boldsymbol{\Theta}} \mathbf{H}_i^{o,T} \right)^T$$

Note that one may also use the standard Karhunen-Loeve decomposition representation of $X_i(t)$ for prediction; see, e.g., Yao et al. (2005). An advantage of the above formulation is that we avoid the evaluation of the eigenfunctions extracted from the covariance function \mathcal{C} ; indeed, we just need to compute the B-spline basis functions at the desired time points, which is computationally simple.

5 Simulations

5.1 Simulation setting

We generate data using model (1). The number of observations for each random curve is generated from a uniform distribution on either $\mathcal{I}_1 = \{3, 4, 5, 6, 7\}$ or $\mathcal{I}_2 = \{j : 5 \leq j \leq 15\}$, and then observations are sampled from a uniform distribution in the unit interval. Therefore, on average, each curve has $m = 5$ or $m = 10$ observations. We use zero mean functions. For the covariance function \mathcal{C} , we consider two cases. For case 1 we let $\mathcal{C}_1(s, t) = \sum_{\ell=1}^3 \lambda_{\ell} \psi_{\ell}(s) \psi_{\ell}(t)$, where ψ_{ℓ} 's are eigenfunctions and λ_{ℓ} 's are eigenvalues. Here $\lambda_{\ell} = 0.5^{\ell-1}$ for $\ell = 1, 2, 3$ and $\psi_1(t) = \sqrt{2} \sin(2\pi t)$, $\psi_2(t) = \sqrt{2} \cos(4\pi t)$ and $\psi_3(t) = \sqrt{2} \sin(4\pi t)$. For case 2 we consider the Matern covariance function

$$C(d; \phi, \nu) = \frac{1}{2^{\nu-1} \Gamma(\nu)} \left(\frac{\sqrt{2\nu} d}{\phi} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu} d}{\phi} \right)$$

with range $\phi = 0.07$ and order $\nu = 1$. Here K_{ν} is the modified Bessel function of order ν . The top two eigenvalues for this covariance function are 0.209 and 0.179, respectively. The noise term ϵ_{ij} 's are assumed normal with mean zero and variance σ_{ϵ}^2 . We consider two levels of signal to noise ratio (SNR): 2 and 5. For example, if

$$\sigma_{\epsilon}^2 = \frac{1}{2} \int_{s=0}^1 \int_{t=0}^1 \mathcal{C}(s, t) ds dt,$$

then the signal to noise ratio in the data is 2. The number of curves is $n = 100$, 200 or 400 and for each covariance function 200 datasets are drawn. Therefore, we have 24 different model conditions to examine.

5.2 Competing methods and evaluation criterion

We compare the proposed method (denoted by FACE) with the following methods: 1) The *fpca.sc* method in Goldsmith et al. (2010), which uses tensor-product bivariate P -splines (Eilers and Marx, 2003) for covariance smoothing and is implemented in the R package *refund*; 2) a variant of *fpca.sc* that uses thin plate regression splines for covariance smoothing, denoted by TPRS,

and is coded by the authors; 3) the MLE method in Peng and Paul (2009), implemented in the R package *fpca*; and 4) the local polynomial method in Yao et al. (2003), denoted by *loc*, and is implemented in the MATLAB toolbox *PACE*. The underlying covariance smoothing functions for *fpca.sc* and TPRS are the function *gam* in the R package *mgcv* (Wood, 2013). For *fpca.sc*, we use its default setting, which uses 10 B-spline bases in each dimension and the smoothing parameters are selected by “REML”. For TPRS, we also use the default setting in *gam*, with the smoothing parameter selected by “REML”. For the method MLE, we specify the range for the number of B-spline bases to be $[6, 10]$ and the range of possible ranks to be $[2, 6]$. We will not evaluate the method using a reduced rank mixed effects model (James et al., 2000) because it has been shown in Peng and Paul (2009) that the MLE method is more superior.

We evaluate the above methods using four criterions. The first is the mean integrated squared errors (MISE) for estimating the covariance function. The next two criterions are based on the eigendecomposition of the covariance function: $\mathcal{C}(s, t) = \sum_{\ell=1}^{\infty} \lambda_{\ell} \psi_{\ell}(s) \psi_{\ell}(t)$, where $\lambda_1 \geq \lambda_2 \geq \dots$ are eigenvalues and $\psi_1(t), \psi_2(t), \dots$ are the associated orthonormal eigenfunctions. The second criterion is the mean integrated squared errors (ISE) for estimating the top 3 eigenfunctions from the covariance function. Let $\psi(t)$ be the true eigenfunction and $\hat{\psi}(t)$ be an estimate of $\psi(t)$, then the mean integrated squared error is

$$\min \left[\int_{t=0}^1 \{\psi(t) - \hat{\psi}(t)\}^2 dt, \int_{t=0}^1 \{\psi(t) + \hat{\psi}(t)\}^2 dt \right].$$

It is easy to show that the range of integrated squared error for eigenfunction estimation is $[0, 2]$. Note that for the method MLE, if rank 2 is selected then only two eigenfunctions can be extracted. In this case, to evaluate accuracy of estimating the third eigenfunction, we will let ISE be 1 for a fair comparison. The third criterion is the mean squared errors (MSE) for estimating the top 3 eigenvalues. The last criterion is the methods’ computation speed.

5.3 Simulation results

The results for estimating the true covariance function are summarized in Table 1, Figure 1 and Figure 2. For most model conditions, FACE gives the smallest medians of integrated squared errors and has the smallest IQRs.

MLE is the 2nd best for case 1 while *loc* is the 2nd best for case 2.

Table 1: Median and IQR (in parenthesis) of ISEs of five estimators for estimating the covariance functions. The results are based on 200 replications.

Case	n	m	SNR	FACE	TPRS	<i>fpca.sc</i>	MLE	<i>loc</i>
Case 1	100	5	2	0.169 (0.085)	0.460 (0.129)	0.305 (0.170)	0.244 (0.224)	0.302 (0.226)
	200	5	2	0.100 (0.050)	0.380 (0.068)	0.206 (0.080)	0.153 (0.109)	0.302 (0.120)
	400	5	2	0.060 (0.028)	0.319 (0.041)	0.122 (0.053)	0.102 (0.077)	0.307 (0.071)
	100	10	2	0.094 (0.050)	0.363 (0.080)	0.184 (0.089)	0.143 (0.129)	0.221 (0.093)
	200	10	2	0.058 (0.032)	0.318 (0.039)	0.108 (0.044)	0.092 (0.082)	0.232 (0.095)
	400	10	2	0.034 (0.019)	0.285 (0.022)	0.057 (0.023)	0.069 (0.050)	0.226 (0.062)
	100	5	5	0.116 (0.070)	0.419 (0.085)	0.255 (0.116)	0.144 (0.125)	0.302 (0.230)
	200	5	5	0.059 (0.032)	0.342 (0.052)	0.155 (0.069)	0.129 (0.075)	0.287 (0.068)
	400	5	5	0.034 (0.017)	0.305 (0.029)	0.094 (0.038)	0.079 (0.060)	0.313 (0.067)
	100	10	5	0.068 (0.056)	0.346 (0.066)	0.154 (0.075)	0.125 (0.094)	0.203 (0.096)
	200	10	5	0.035 (0.022)	0.302 (0.033)	0.086 (0.036)	0.080 (0.073)	0.218 (0.072)
	400	10	5	0.018 (0.011)	0.280 (0.017)	0.047 (0.021)	0.063 (0.042)	0.229 (0.062)
Case 2	100	5	2	0.047 (0.017)	0.061 (0.024)	0.070 (0.040)	0.090 (0.053)	0.049 (0.015)
	200	5	2	0.029 (0.011)	0.044 (0.015)	0.046 (0.020)	0.045 (0.025)	0.038 (0.010)
	400	5	2	0.019 (0.006)	0.031 (0.009)	0.030 (0.015)	0.028 (0.010)	0.029 (0.006)
	100	10	2	0.025 (0.010)	0.041 (0.013)	0.047 (0.022)	0.040 (0.017)	0.034 (0.009)
	200	10	2	0.016 (0.005)	0.030 (0.007)	0.028 (0.011)	0.024 (0.009)	0.025 (0.006)
	400	10	2	0.009 (0.003)	0.022 (0.003)	0.016 (0.005)	0.015 (0.004)	0.021 (0.004)
	100	5	5	0.038 (0.017)	0.054 (0.020)	0.060 (0.030)	0.066 (0.038)	0.043 (0.014)
	200	5	5	0.022 (0.008)	0.038 (0.012)	0.038 (0.019)	0.037 (0.015)	0.033 (0.009)
	400	5	5	0.014 (0.004)	0.027 (0.006)	0.024 (0.009)	0.023 (0.008)	0.027 (0.004)
	100	10	5	0.020 (0.006)	0.035 (0.010)	0.038 (0.016)	0.033 (0.012)	0.032 (0.009)
	200	10	5	0.012 (0.004)	0.025 (0.006)	0.022 (0.007)	0.019 (0.006)	0.024 (0.005)
	400	10	5	0.007 (0.003)	0.020 (0.003)	0.013 (0.004)	0.013 (0.003)	0.020 (0.003)

Table 2 reports the results for estimating the 1st eigenfunction. The re-

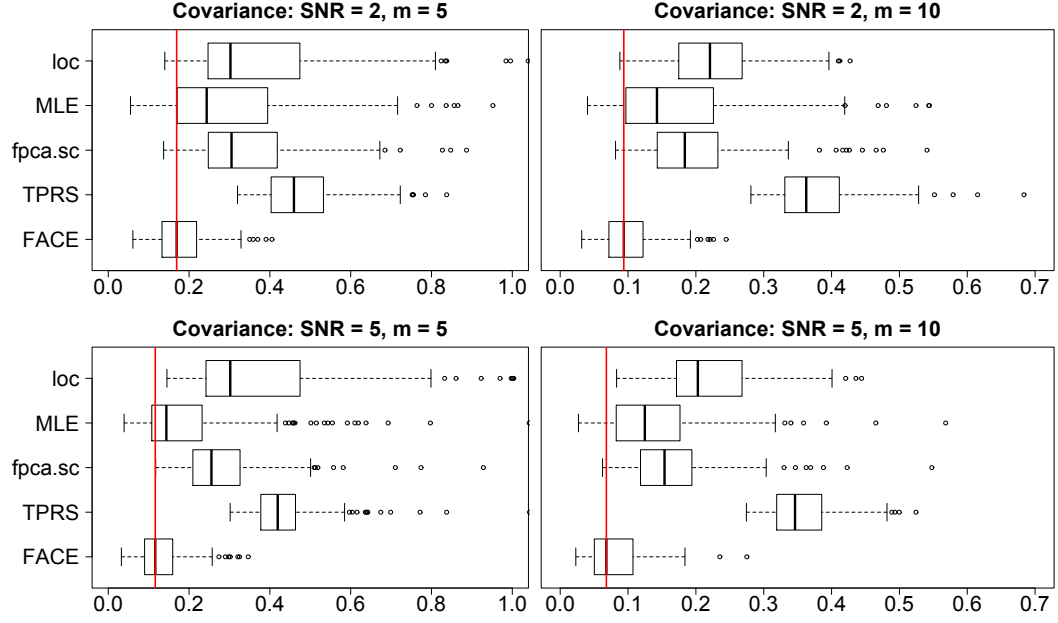


Figure 1: Boxplots of ISEs of five estimators for estimating the covariance functions of case 1, $n = 100$.

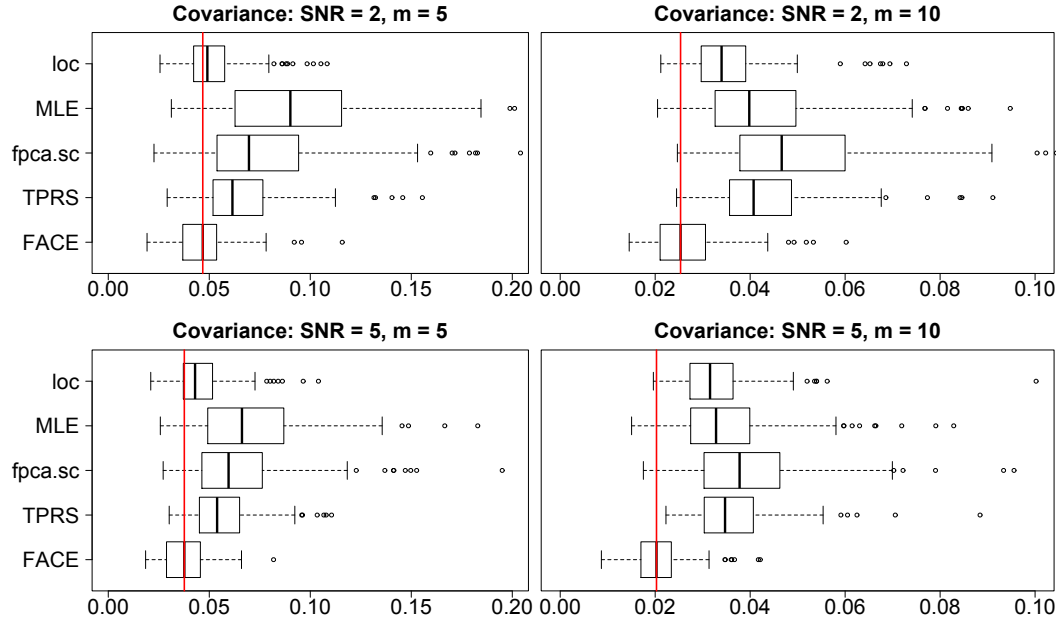


Figure 2: Boxplots of ISEs of five estimators for estimating the covariance functions of case 2, $n = 100$.

sults for estimating the 2nd and 3rd eigenfunctions are shown in Table 3 and Table 4, respectively. Figure 3 illustrates the superiority of FACE for estimating eigenfunctions when $n = 100$, $m = 5$. FACE tends to outperform other approaches in most scenarios, while for the remaining scenarios, its performance is still comparable with the best one. MLE performs well for case 1 but relatively poorly for case 2, while the opposite is true for *loc*. TPRS and *fpca.sc* perform quite poorly for estimating the 2nd and 3rd eigenfunctions in both case 1 and case 2.

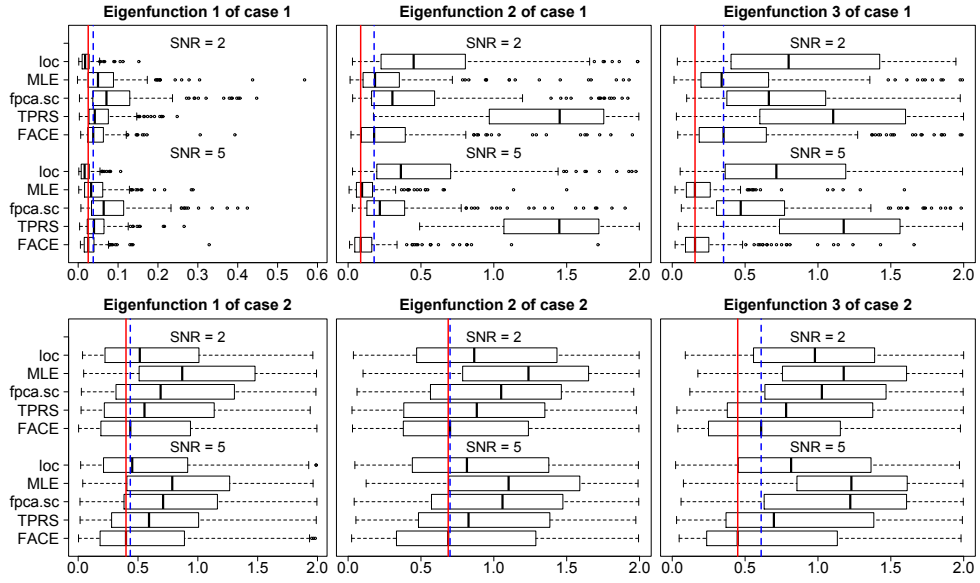


Figure 3: Boxplots of ISEs of five estimators for estimating the top 3 eigenfunctions when $n = 100$, $m = 5$. Note that the straight lines are the medians of FACE when $SNR = 5$ and the dash lines are the medians of FACE when $SNR = 2$.

The results for estimation of eigenvalues are shown in Tables 5, 6, and 7. We have the following findings: 1) FACE performs the best for estimating the first eigenvalue in case 1; 2) *loc* performs the best for estimating the first eigenvalue in case 2; 3) MLE performs overall the best for estimating 2nd and 3rd eigenvalues in both cases, while the performance of FACE is very close and can be better than MLE under some model scenarios; 4) TPRS, *fpca.sc* and *loc* perform quite poorly for estimating the 2nd and 3rd eigenvalues in most scenarios. We conclude that FACE shows overall very competitive

Table 2: Median and IQR (in parenthesis) of ISEs of five estimators for estimating the 1st eigenfunction. The results are based on 200 replications.

Case	n	m	SNR	FACE	TPRS	<i>fpca.sc</i>	MLE	<i>loc</i>
Case 1	100	5	2	0.038 (0.039)	0.042 (0.048)	0.071 (0.092)	0.050 (0.064)	0.017 (0.017)
	200	5	2	0.024 (0.026)	0.029 (0.026)	0.051 (0.063)	0.030 (0.029)	0.026 (0.026)
	400	5	2	0.013 (0.011)	0.015 (0.014)	0.025 (0.024)	0.012 (0.012)	0.018 (0.016)
	100	10	2	0.020 (0.026)	0.026 (0.025)	0.058 (0.062)	0.029 (0.034)	0.025 (0.025)
	200	10	2	0.015 (0.015)	0.016 (0.013)	0.030 (0.030)	0.012 (0.011)	0.015 (0.014)
	400	10	2	0.008 (0.009)	0.009 (0.007)	0.016 (0.016)	0.005 (0.007)	0.010 (0.008)
	100	5	5	0.025 (0.024)	0.040 (0.041)	0.064 (0.080)	0.032 (0.046)	0.016 (0.020)
	200	5	5	0.013 (0.013)	0.024 (0.019)	0.041 (0.042)	0.019 (0.019)	0.022 (0.019)
	400	5	5	0.009 (0.010)	0.013 (0.010)	0.023 (0.024)	0.009 (0.010)	0.018 (0.016)
	100	10	5	0.014 (0.017)	0.024 (0.025)	0.052 (0.060)	0.025 (0.028)	0.021 (0.025)
	200	10	5	0.008 (0.009)	0.013 (0.010)	0.025 (0.025)	0.011 (0.011)	0.012 (0.013)
	400	10	5	0.005 (0.006)	0.009 (0.007)	0.014 (0.015)	0.004 (0.004)	0.008 (0.008)
Case 2	100	5	2	0.436 (0.748)	0.554 (0.916)	0.689 (0.981)	0.869 (0.959)	0.515 (0.781)
	200	5	2	0.338 (0.722)	0.422 (0.794)	0.557 (0.854)	0.501 (0.808)	0.317 (0.573)
	400	5	2	0.329 (0.556)	0.434 (0.698)	0.464 (0.730)	0.398 (0.718)	0.209 (0.455)
	100	10	2	0.437 (0.610)	0.513 (0.709)	0.645 (0.713)	0.578 (0.696)	0.408 (0.638)
	200	10	2	0.312 (0.649)	0.378 (0.690)	0.470 (0.743)	0.417 (0.682)	0.208 (0.375)
	400	10	2	0.205 (0.517)	0.236 (0.491)	0.272 (0.516)	0.217 (0.457)	0.154 (0.333)
	100	5	5	0.400 (0.701)	0.592 (0.720)	0.710 (0.777)	0.786 (0.859)	0.452 (0.701)
	200	5	5	0.427 (0.627)	0.462 (0.759)	0.636 (0.792)	0.520 (0.815)	0.329 (0.625)
	400	5	5	0.225 (0.496)	0.329 (0.477)	0.396 (0.644)	0.280 (0.604)	0.167 (0.271)
	100	10	5	0.390 (0.643)	0.499 (0.735)	0.593 (0.828)	0.468 (0.668)	0.362 (0.666)
	200	10	5	0.217 (0.445)	0.317 (0.537)	0.375 (0.511)	0.282 (0.448)	0.184 (0.401)
	400	10	5	0.161 (0.327)	0.209 (0.323)	0.248 (0.327)	0.147 (0.298)	0.095 (0.210)

Table 3: Median and IQR (in parenthesis) of ISEs of five estimators for estimating the 2nd eigenfunction. The results are based on 200 replications.

Case	n	m	SNR	FACE	TPRS	<i>fpca.sc</i>	MLE	<i>loc</i>
Case 1	100	5	2	0.179 (0.300)	1.452 (0.783)	0.304 (0.427)	0.185 (0.249)	0.450 (0.573)
	200	5	2	0.072 (0.085)	1.477 (0.641)	0.166 (0.142)	0.067 (0.078)	0.218 (0.291)
	400	5	2	0.037 (0.037)	1.623 (0.468)	0.079 (0.071)	0.036 (0.034)	0.146 (0.127)
	100	10	2	0.069 (0.079)	1.505 (0.583)	0.141 (0.147)	0.070 (0.089)	0.133 (0.136)
	200	10	2	0.035 (0.040)	1.663 (0.433)	0.077 (0.057)	0.032 (0.040)	0.094 (0.075)
	400	10	2	0.017 (0.021)	1.738 (0.323)	0.037 (0.036)	0.015 (0.014)	0.073 (0.047)
	100	5	5	0.086 (0.117)	1.449 (0.647)	0.218 (0.258)	0.096 (0.111)	0.362 (0.505)
	200	5	5	0.035 (0.045)	1.498 (0.563)	0.111 (0.103)	0.045 (0.042)	0.187 (0.172)
	400	5	5	0.020 (0.024)	1.675 (0.419)	0.061 (0.063)	0.021 (0.022)	0.124 (0.107)
	100	10	5	0.043 (0.058)	1.560 (0.475)	0.101 (0.114)	0.050 (0.051)	0.107 (0.120)
	200	10	5	0.019 (0.023)	1.668 (0.448)	0.059 (0.048)	0.019 (0.025)	0.084 (0.059)
	400	10	5	0.011 (0.012)	1.742 (0.309)	0.032 (0.030)	0.009 (0.009)	0.070 (0.036)
Case 2	100	5	2	0.700 (0.856)	0.883 (0.967)	1.050 (0.892)	1.238 (0.862)	0.865 (0.959)
	200	5	2	0.682 (0.992)	0.771 (0.875)	1.013 (0.935)	1.031 (0.931)	0.655 (0.867)
	400	5	2	0.543 (0.823)	0.701 (0.946)	0.838 (0.962)	0.828 (0.782)	0.450 (0.686)
	100	10	2	0.640 (0.985)	0.779 (0.887)	0.969 (0.960)	0.971 (0.941)	0.729 (0.860)
	200	10	2	0.601 (0.920)	0.688 (0.903)	0.765 (0.887)	0.758 (0.899)	0.430 (0.552)
	400	10	2	0.374 (0.658)	0.438 (0.785)	0.508 (0.760)	0.374 (0.583)	0.271 (0.424)
	100	5	5	0.687 (0.951)	0.826 (0.901)	1.060 (0.901)	1.101 (0.900)	0.815 (0.935)
	200	5	5	0.718 (0.932)	0.777 (1.028)	0.893 (0.983)	0.939 (0.897)	0.669 (0.903)
	400	5	5	0.449 (0.676)	0.587 (0.704)	0.706 (0.841)	0.624 (0.934)	0.364 (0.425)
	100	10	5	0.701 (0.963)	0.876 (0.986)	1.035 (0.906)	0.881 (0.878)	0.723 (0.861)
	200	10	5	0.426 (0.585)	0.589 (0.913)	0.682 (0.944)	0.513 (0.636)	0.392 (0.629)
	400	10	5	0.257 (0.446)	0.340 (0.492)	0.384 (0.579)	0.264 (0.437)	0.203 (0.288)

Table 4: Median and IQR (in parenthesis) of ISEs of five estimators for estimating the 3rd eigenfunction. The results are based on 200 replications.

Case	n	m	SNR	FACE	TPRS	<i>fpca.sc</i>	MLE	<i>loc</i>
Case 1	100	5	2	0.352 (0.455)	1.104 (0.998)	0.663 (0.670)	0.339 (0.462)	0.799 (1.020)
	200	5	2	0.165 (0.156)	1.210 (0.920)	0.332 (0.346)	0.175 (0.161)	0.468 (0.621)
	400	5	2	0.069 (0.061)	1.527 (0.553)	0.163 (0.114)	0.057 (0.058)	0.229 (0.248)
	100	10	2	0.110 (0.124)	1.425 (0.784)	0.226 (0.165)	0.117 (0.120)	0.200 (0.251)
	200	10	2	0.064 (0.065)	1.556 (0.558)	0.118 (0.088)	0.043 (0.042)	0.125 (0.126)
	400	10	2	0.035 (0.040)	1.740 (0.355)	0.060 (0.044)	0.021 (0.017)	0.078 (0.070)
	100	5	5	0.156 (0.160)	1.177 (0.813)	0.470 (0.466)	0.156 (0.164)	0.715 (0.827)
	200	5	5	0.063 (0.054)	1.369 (0.708)	0.198 (0.179)	0.075 (0.072)	0.309 (0.363)
	400	5	5	0.036 (0.029)	1.558 (0.505)	0.119 (0.089)	0.027 (0.025)	0.197 (0.194)
	100	10	5	0.054 (0.066)	1.489 (0.629)	0.150 (0.143)	0.056 (0.061)	0.148 (0.188)
	200	10	5	0.026 (0.026)	1.629 (0.532)	0.085 (0.078)	0.021 (0.026)	0.096 (0.090)
	400	10	5	0.014 (0.013)	1.723 (0.372)	0.046 (0.030)	0.010 (0.009)	0.067 (0.052)
Case 2	100	5	2	0.610 (0.900)	0.781 (0.992)	1.027 (0.823)	1.177 (0.852)	0.979 (0.827)
	200	5	2	0.551 (0.989)	0.679 (0.751)	1.012 (0.760)	1.187 (0.849)	0.747 (0.811)
	400	5	2	0.363 (0.506)	0.511 (0.728)	0.760 (0.795)	0.759 (0.795)	0.408 (0.489)
	100	10	2	0.663 (0.952)	0.721 (0.917)	1.111 (0.869)	1.093 (0.907)	0.744 (0.826)
	200	10	2	0.470 (0.595)	0.590 (0.746)	0.936 (0.919)	0.704 (0.680)	0.431 (0.494)
	400	10	2	0.308 (0.471)	0.319 (0.491)	0.505 (0.704)	0.392 (0.598)	0.279 (0.242)
	100	5	5	0.449 (0.894)	0.698 (1.010)	1.222 (0.972)	1.229 (0.757)	0.816 (0.910)
	200	5	5	0.614 (0.700)	0.670 (0.905)	1.004 (0.866)	1.052 (0.841)	0.588 (0.588)
	400	5	5	0.321 (0.582)	0.425 (0.646)	0.709 (0.698)	0.626 (0.725)	0.372 (0.362)
	100	10	5	0.812 (0.898)	0.791 (0.858)	1.096 (0.789)	1.121 (0.846)	0.788 (0.823)
	200	10	5	0.371 (0.532)	0.501 (0.630)	0.654 (0.759)	0.535 (0.657)	0.359 (0.365)
	400	10	5	0.193 (0.252)	0.267 (0.375)	0.362 (0.455)	0.257 (0.300)	0.219 (0.199)

performance and never deviates much from the best performance. We use Figure 4 to illustrate the patterns of eigenvalue estimation for $n = 100$, $m = 5$.

Table 5: $100\times$ Median and IQR (in parenthesis) of SEs of five estimators for estimating the 1st eigenvalue. The results are based on 200 replications.

Case	n	m	SNR	FACE	TPRS	<i>fpca.sc</i>	MLE	<i>loc</i>
Case 1	100	5	2	1.736 (5.183)	2.914 (6.793)	2.335 (6.261)	3.877 (9.450)	5.749 (21.780)
	200	5	2	0.624 (2.112)	1.507 (3.510)	1.057 (2.680)	4.502 (7.193)	9.324 (9.992)
	400	5	2	0.501 (1.213)	0.624 (1.742)	0.603 (1.510)	4.911 (6.375)	11.312 (6.953)
	100	10	2	1.212 (3.228)	1.772 (5.345)	1.486 (4.460)	4.009 (9.774)	4.366 (8.535)
	200	10	2	0.795 (1.776)	1.210 (2.502)	1.250 (2.418)	4.336 (8.115)	7.887 (7.807)
	400	10	2	0.258 (0.700)	0.290 (0.792)	0.284 (0.845)	4.840 (4.729)	8.184 (4.808)
	100	5	5	2.087 (4.322)	1.945 (5.038)	1.463 (4.701)	2.708 (7.754)	5.927 (23.788)
	200	5	5	0.550 (2.008)	0.774 (2.252)	0.657 (1.847)	5.297 (6.989)	8.499 (6.500)
	400	5	5	0.370 (0.855)	0.602 (1.434)	0.540 (1.165)	4.783 (5.177)	11.382 (5.761)
	100	10	5	1.387 (3.910)	1.582 (4.156)	1.337 (3.598)	4.295 (8.905)	4.576 (7.862)
	200	10	5	0.523 (1.706)	0.778 (2.150)	0.763 (2.105)	4.904 (6.767)	7.577 (7.104)
	400	10	5	0.192 (0.534)	0.311 (0.721)	0.308 (0.690)	4.888 (4.324)	8.222 (5.181)
Case 2	100	5	2	0.265 (0.716)	0.567 (1.193)	0.816 (1.530)	1.359 (1.865)	0.211 (0.542)
	200	5	2	0.132 (0.301)	0.245 (0.734)	0.341 (0.904)	0.447 (0.693)	0.124 (0.294)
	400	5	2	0.071 (0.185)	0.104 (0.312)	0.156 (0.435)	0.180 (0.281)	0.047 (0.098)
	100	10	2	0.115 (0.269)	0.241 (0.556)	0.417 (0.786)	0.307 (0.601)	0.084 (0.273)
	200	10	2	0.057 (0.176)	0.111 (0.299)	0.146 (0.384)	0.112 (0.277)	0.062 (0.147)
	400	10	2	0.032 (0.068)	0.046 (0.092)	0.056 (0.113)	0.045 (0.093)	0.026 (0.065)
	100	5	5	0.127 (0.405)	0.331 (0.840)	0.537 (1.242)	0.783 (1.289)	0.134 (0.437)
	200	5	5	0.076 (0.201)	0.162 (0.418)	0.275 (0.543)	0.236 (0.462)	0.062 (0.152)
	400	5	5	0.034 (0.105)	0.069 (0.202)	0.095 (0.254)	0.113 (0.229)	0.033 (0.086)
	100	10	5	0.063 (0.194)	0.154 (0.383)	0.225 (0.503)	0.207 (0.483)	0.074 (0.225)
	200	10	5	0.039 (0.116)	0.077 (0.192)	0.095 (0.248)	0.087 (0.186)	0.042 (0.102)
	400	10	5	0.028 (0.080)	0.031 (0.081)	0.038 (0.106)	0.046 (0.091)	0.023 (0.052)

Finally, Table 8 and Figure 5 summarize the run times. When $m = 5$, FACE takes about three to six times the computation times of TPRS and *fpca.sc*; but it is much faster than MLE and *loc*, the speed-up is about 20 and 70 fold, respectively. When $m = 10$, the run times of FACE and *fpca.sc* are

Table 6: $100\times$ Median and IQR (in parenthesis) of SEs of five estimators for estimating the 2nd eigenvalue. The results are based on 200 replications.

Case	n	m	SNR	FACE	TPRS	<i>fpca.sc</i>	MLE	<i>loc</i>
Case 1	100	5	2	1.912 (2.943)	9.879 (6.997)	1.530 (3.118)	0.486 (1.564)	9.088 (4.855)
	200	5	2	0.659 (1.710)	10.064 (5.502)	0.783 (1.760)	0.324 (0.857)	10.906 (3.227)
	400	5	2	0.313 (0.889)	9.780 (3.503)	0.451 (0.927)	0.146 (0.304)	11.891 (2.358)
	100	10	2	0.869 (1.573)	9.662 (4.534)	0.565 (1.554)	0.261 (0.736)	7.351 (3.697)
	200	10	2	0.299 (0.978)	9.622 (3.784)	0.282 (0.740)	0.172 (0.488)	8.276 (3.345)
	400	10	2	0.205 (0.444)	9.901 (2.740)	0.152 (0.435)	0.096 (0.273)	8.810 (2.439)
	100	5	5	2.079 (2.585)	10.684 (5.510)	1.677 (3.346)	0.455 (1.420)	9.634 (5.118)
	200	5	5	0.908 (1.556)	10.065 (4.927)	0.628 (1.560)	0.216 (0.583)	11.419 (2.904)
	400	5	5	0.356 (0.769)	10.164 (3.484)	0.288 (0.762)	0.100 (0.292)	12.017 (1.920)
	100	10	5	1.795 (1.835)	10.207 (4.581)	0.492 (1.327)	0.242 (0.681)	7.831 (4.057)
	200	10	5	0.723 (0.965)	9.767 (3.036)	0.214 (0.599)	0.166 (0.453)	8.338 (2.999)
	400	10	5	0.311 (0.441)	9.793 (1.964)	0.123 (0.295)	0.055 (0.175)	9.106 (2.132)
Case 2	100	5	2	0.165 (0.333)	0.177 (0.447)	0.174 (0.430)	0.241 (0.596)	0.140 (0.333)
	200	5	2	0.042 (0.112)	0.053 (0.134)	0.057 (0.200)	0.072 (0.211)	0.081 (0.187)
	400	5	2	0.021 (0.060)	0.027 (0.065)	0.027 (0.081)	0.035 (0.077)	0.098 (0.196)
	100	10	2	0.063 (0.158)	0.061 (0.146)	0.067 (0.207)	0.046 (0.144)	0.091 (0.212)
	200	10	2	0.023 (0.062)	0.030 (0.094)	0.033 (0.090)	0.022 (0.060)	0.083 (0.174)
	400	10	2	0.015 (0.035)	0.012 (0.050)	0.014 (0.054)	0.015 (0.040)	0.056 (0.121)
	100	5	5	0.079 (0.252)	0.128 (0.316)	0.089 (0.270)	0.136 (0.302)	0.154 (0.367)
	200	5	5	0.024 (0.102)	0.064 (0.166)	0.065 (0.169)	0.049 (0.141)	0.121 (0.233)
	400	5	5	0.016 (0.042)	0.029 (0.071)	0.029 (0.069)	0.028 (0.069)	0.110 (0.216)
	100	10	5	0.041 (0.102)	0.043 (0.115)	0.043 (0.125)	0.038 (0.098)	0.115 (0.211)
	200	10	5	0.015 (0.044)	0.031 (0.066)	0.031 (0.069)	0.017 (0.043)	0.087 (0.161)
	400	10	5	0.012 (0.037)	0.010 (0.034)	0.012 (0.033)	0.011 (0.036)	0.068 (0.126)

Table 7: $100\times$ Median and IQR (in parenthesis) of SEs of five estimators for estimating the 3rd eigenvalue. The results are based on 200 replications.

Case	n	m	SNR	FACE	TPRS	<i>fpca.sc</i>	MLE	<i>loc</i>
Case 1	100	5	2	0.167 (0.491)	3.147 (2.115)	0.500 (1.095)	0.224 (0.711)	2.521 (1.915)
	200	5	2	0.135 (0.276)	2.843 (2.330)	0.250 (0.710)	0.132 (0.383)	2.763 (1.365)
	400	5	2	0.077 (0.197)	2.778 (1.909)	0.157 (0.406)	0.088 (0.218)	2.986 (0.959)
	100	10	2	0.115 (0.325)	2.964 (1.778)	0.285 (0.552)	0.114 (0.371)	2.337 (1.448)
	200	10	2	0.055 (0.137)	3.019 (1.679)	0.162 (0.331)	0.062 (0.203)	2.487 (0.934)
	400	10	2	0.037 (0.084)	3.113 (1.452)	0.094 (0.186)	0.028 (0.087)	2.581 (0.833)
	100	5	5	0.274 (0.537)	3.237 (2.045)	0.453 (0.914)	0.192 (0.518)	2.646 (1.684)
	200	5	5	0.070 (0.203)	2.801 (2.020)	0.150 (0.459)	0.080 (0.221)	2.860 (1.167)
	400	5	5	0.034 (0.084)	3.049 (1.690)	0.122 (0.290)	0.038 (0.086)	3.066 (0.808)
	100	10	5	0.130 (0.288)	3.059 (1.669)	0.193 (0.427)	0.093 (0.224)	2.416 (1.112)
	200	10	5	0.037 (0.127)	3.038 (1.704)	0.111 (0.245)	0.033 (0.118)	2.565 (0.879)
	400	10	5	0.019 (0.054)	2.960 (1.349)	0.038 (0.116)	0.021 (0.063)	2.680 (0.590)
Case 2	100	5	2	0.418 (1.067)	0.407 (0.826)	0.184 (0.501)	0.085 (0.230)	0.370 (0.474)
	200	5	2	0.093 (0.283)	0.140 (0.373)	0.059 (0.163)	0.029 (0.083)	0.286 (0.373)
	400	5	2	0.029 (0.083)	0.056 (0.138)	0.030 (0.067)	0.016 (0.039)	0.258 (0.311)
	100	10	2	0.072 (0.154)	0.097 (0.188)	0.035 (0.100)	0.019 (0.065)	0.227 (0.294)
	200	10	2	0.030 (0.078)	0.047 (0.102)	0.025 (0.065)	0.016 (0.049)	0.196 (0.248)
	400	10	2	0.012 (0.032)	0.018 (0.048)	0.013 (0.037)	0.011 (0.027)	0.162 (0.151)
	100	5	5	0.204 (0.547)	0.257 (0.478)	0.092 (0.294)	0.060 (0.203)	0.380 (0.443)
	200	5	5	0.046 (0.140)	0.119 (0.279)	0.056 (0.142)	0.029 (0.069)	0.291 (0.321)
	400	5	5	0.013 (0.041)	0.039 (0.090)	0.019 (0.051)	0.017 (0.048)	0.224 (0.236)
	100	10	5	0.044 (0.102)	0.067 (0.149)	0.026 (0.072)	0.023 (0.054)	0.217 (0.251)
	200	10	5	0.016 (0.037)	0.031 (0.063)	0.017 (0.042)	0.011 (0.032)	0.174 (0.208)
	400	10	5	0.010 (0.028)	0.013 (0.037)	0.014 (0.029)	0.010 (0.025)	0.154 (0.159)

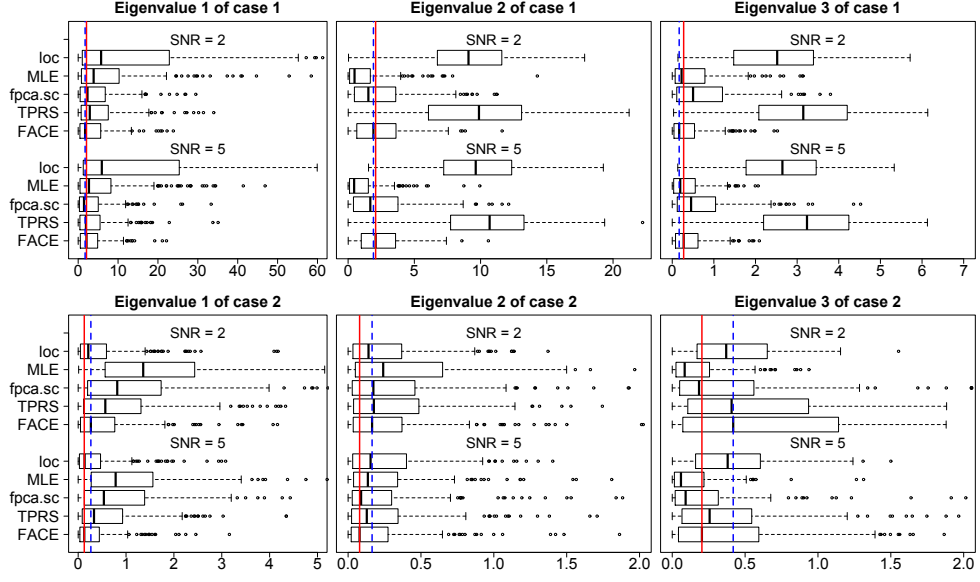


Figure 4: Boxplots of $100 \times$ SEs of five estimators for estimating the eigenfunctions when $n = 100$, $m = 5$. Note that the straight lines are the medians of FACE when $SNR = 5$ and the dash lines are the medians of FACE when $SNR = 2$.

quite close to each other, and FACE is only slower than TPRS; run times of MLE and *loc* are over 13 and 160 fold of FACE, respectively. Because TPRS and *fpca.sc* are naive covariance smoothers, their fast speed is offset by their tendency to have inferior performance in terms of estimation of covariance functions, eigenfunctions, and eigenvalues.

To summarize, FACE is a relatively fast method coupled with competing performance against the methods examined above.

6 Applications

6.1 Child growth data

The Contents study was conducted in Pampas de San Juan Miraflores and Nuevo Paraso, two peri-urban shanty towns with high population density, 25 km south of central Lima. These peri-urban communities are comprised of 50,000 residents, the majority of whom are immigrants from rural areas of the Peruvian Andes who settled nearly 35 years ago and later claimed unused

Table 8: Median and IQR (in parenthesis) of computation times (in seconds) of five estimators for estimating the covariance functions on a desktop with a 2.3 GHz CPU and 8 GB of RAM. The results are based on 200 replications.

Case	n	m	SNR	FACE	TPRS	<i>fpca.sc</i>	MLE	<i>loc</i>
Case 1	100	5	2	7.900 (0.100)	4.100 (1.200)	2.400 (1.200)	141.700 (52.700)	477.300 (43.800)
	200	5	2	15.400 (0.600)	7.000 (0.300)	7.000 (0.500)	397.700 (52.000)	787.300 (407.100)
	400	5	2	30.800 (6.700)	8.500 (2.900)	11.000 (7.100)	615.100 (209.600)	1304.800 (239.000)
	100	10	2	14.200 (1.000)	7.500 (1.400)	12.200 (4.400)	153.100 (41.700)	1680.500 (258.900)
	200	10	2	27.900 (1.100)	10.300 (0.700)	18.800 (2.700)	249.200 (35.100)	315.900 (39.600)
	400	10	2	54.600 (4.900)	23.300 (4.000)	55.700 (17.500)	779.800 (149.300)	772.800 (31.500)
	100	5	5	9.900 (0.300)	4.600 (0.200)	2.900 (0.300)	174.200 (26.000)	462.500 (83.800)
	200	5	5	19.200 (3.400)	5.500 (0.200)	5.000 (0.300)	295.900 (44.800)	1779.900 (355.500)
	400	5	5	32.000 (8.400)	10.000 (2.000)	14.100 (4.500)	767.300 (216.400)	2462.800 (387.200)
	100	10	5	14.700 (0.800)	6.900 (1.500)	10.900 (3.300)	143.900 (46.500)	2781.000 (426.800)
	200	10	5	28.700 (1.500)	12.100 (3.700)	23.800 (7.500)	361.600 (107.000)	344.400 (26.400)
	400	10	5	45.800 (4.400)	23.600 (4.900)	55.600 (17.400)	758.300 (137.200)	486.000 (36.700)
Case 2	100	5	2	8.900 (4.800)	4.100 (0.300)	2.300 (0.300)	166.600 (28.700)	1035.500 (112.500)
	200	5	2	16.500 (0.700)	6.900 (0.500)	6.800 (0.900)	455.100 (51.500)	1173.200 (135.900)
	400	5	2	37.100 (12.100)	8.900 (2.400)	11.400 (4.100)	783.900 (262.200)	2704.500 (1303.500)
	100	10	2	13.200 (0.700)	6.200 (0.600)	9.600 (2.100)	150.300 (23.100)	2302.800 (461.400)
	200	10	2	22.400 (1.400)	10.900 (0.500)	20.400 (1.400)	368.300 (33.100)	203.300 (50.200)
	400	10	2	52.200 (16.700)	19.500 (3.700)	42.700 (13.700)	701.200 (117.800)	362.800 (63.300)
	100	5	5	9.500 (0.300)	4.100 (0.200)	2.400 (0.200)	179.800 (17.800)	739.400 (60.100)
	200	5	5	17.900 (2.400)	6.200 (1.200)	5.900 (1.400)	420.000 (104.600)	1195.800 (356.300)
	400	5	5	29.600 (4.000)	9.000 (0.600)	13.400 (1.700)	835.100 (97.700)	2041.200 (981.400)
	100	10	5	13.400 (0.900)	6.900 (0.300)	11.300 (1.100)	172.000 (13.000)	2719.900 (1005.600)
	200	10	5	27.200 (1.800)	10.400 (3.000)	22.900 (8.200)	311.500 (128.100)	1601.700 (171.800)
	400	10	5	45.300 (5.500)	22.100 (5.300)	52.900 (17.800)	772.800 (210.800)	1532.400 (247.800)

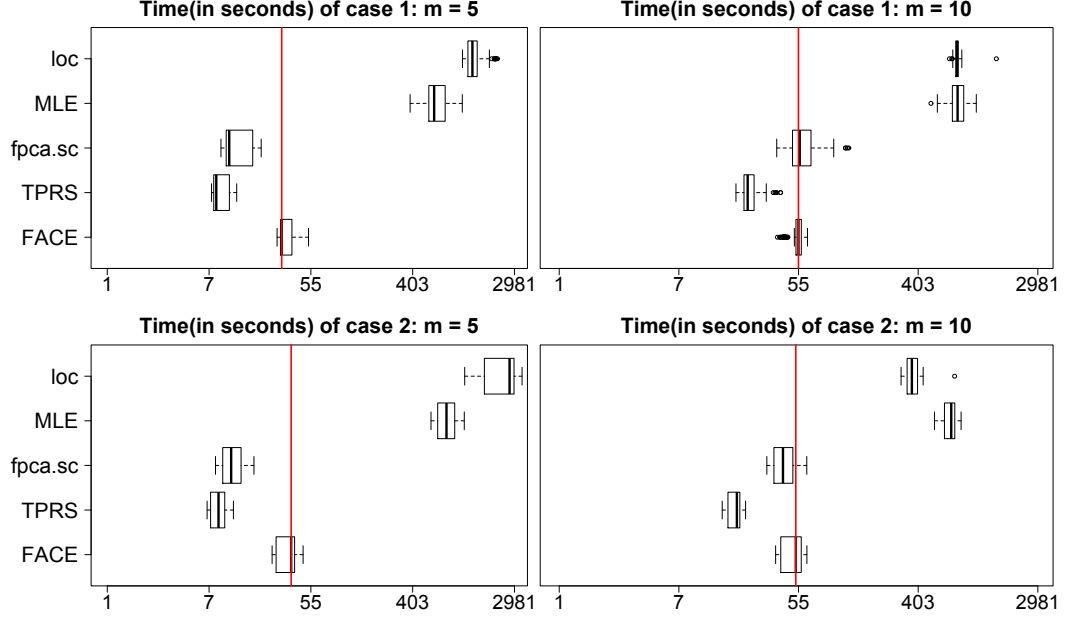


Figure 5: Boxplots of computation times (in seconds) of five estimators for estimating the covariance functions when $n = 400$, $SNR = 2$.

land on the outskirts of Lima. In the last two decades Pampas has undergone many economic and social developments. The study contains 197 children with anthropomorphic measurements taken from birth. Here we focus on the length curves from birth to 1 year. Each child has 10 to 32 measurements of length, with 4320 data points in total. Figure 6 displays the sample length trajectories of 4 children. We apply the proposed method to the data. The estimated population mean is plotted in Figure 6 as a solid line. The estimated mean curve is generally increasing with age, which is expected. In Figure 7, we plot the estimated variance and correlation functions. The estimated variance function is increasing as a function of age. The estimated correlation function is shown as a heat map. Each point in the heat map represents the estimated correlation between two days and the color corresponds to the correlation values with red indicating higher correlation and blue indicating lower correlation. The diagonal is dark red indicating perfect correlation, while the minimum correlation is about 0.2. Given the estimated covariance function, using the framework described in Section 4, we predict each child's length

trajectories. For the 4 children in Figure 6, Figure 8 displays the predicted length trajectories with point-wise confidence bands. We see that the point-wise confidence intervals are very narrow due to that for these 4 children a large number of observations are available.

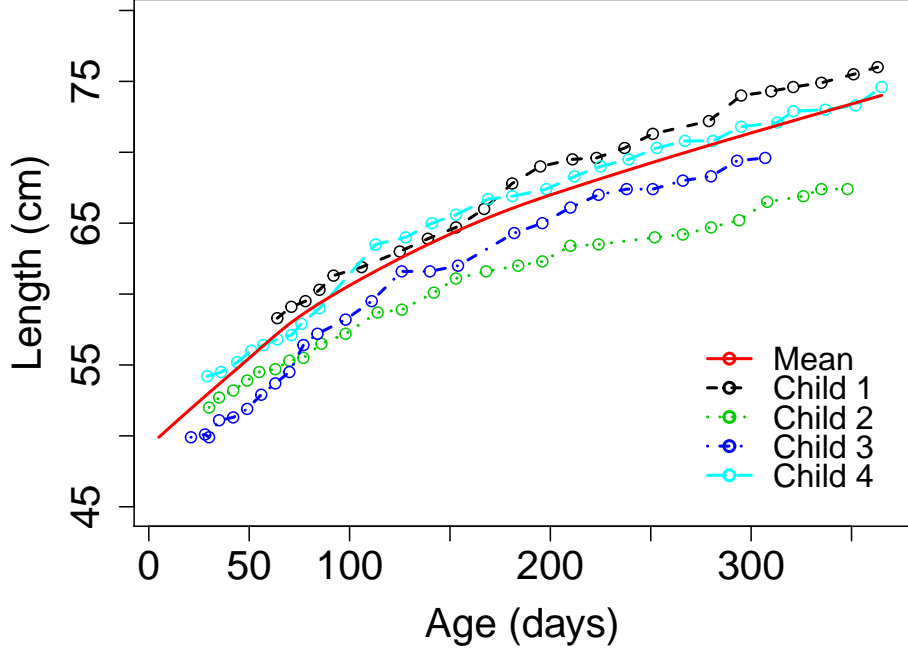


Figure 6: Length trajectories of 4 children from birth to 1 year old. The estimated population mean is the solid line.

6.2 CD4 data

CD4 cells are a type of white blood cells that could send signals to the human body to activate the immune response when they detect viruses or bacteria. Thus, the CD4 count is an important biomarker used for assessing the health of HIV infected persons as HIV viruses attack and destroy the CD4 cells. The dataset analyzed here is from the Multicenter AIDS Cohort Study (MACS) and is available in the *refund* R package (Huang et al., 2015). The observations are CD4 cell counts for 366 infected males in a longitudinal study (Kaslow et al., 1987). With a total of 1888 data points, each subject has be-

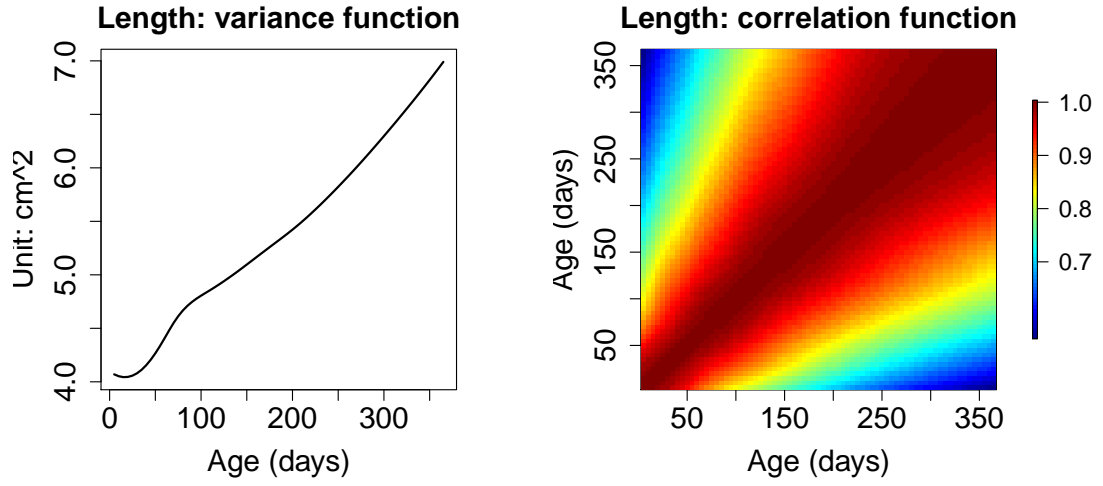


Figure 7: Estimated variance function (left panel) and correlation function (right panel) for the length of children from birth to 1 year old.

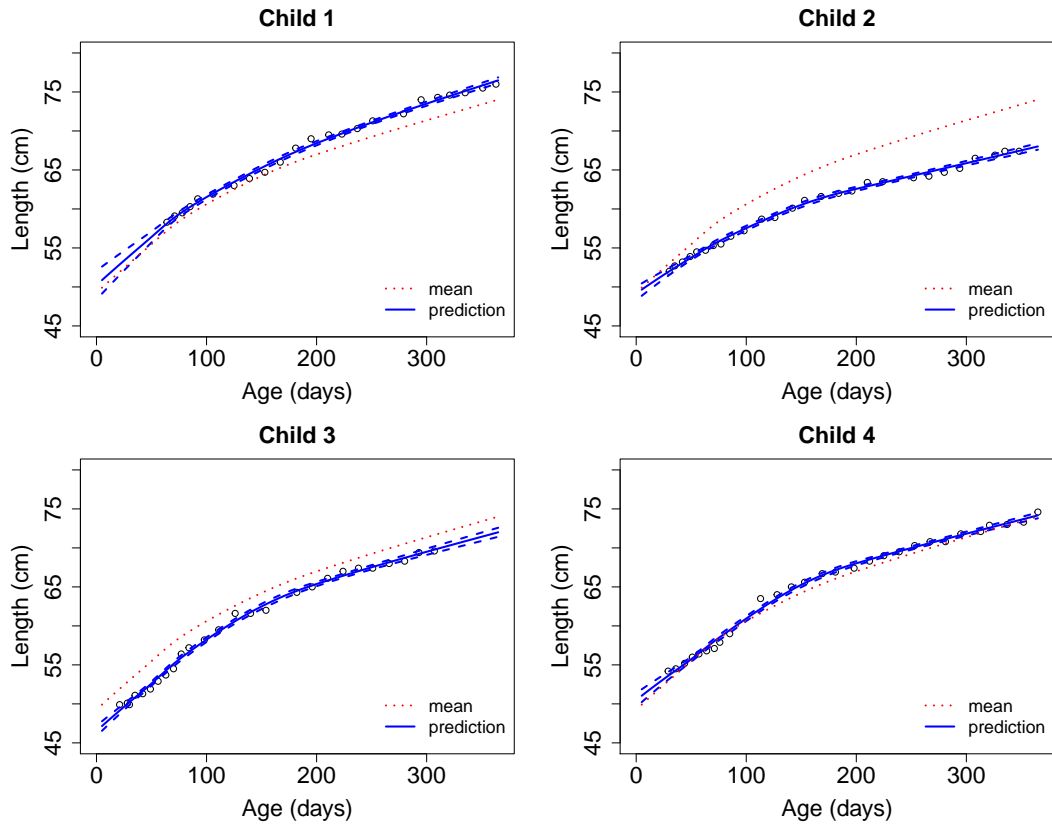


Figure 8: Predicted child-specific length trajectories from birth to 1 year old and associated 95% confidence bands for 4 children. The estimated population mean is the dotted line.

tween 1 and 11 observations. Statistical analysis based on this or related datasets were done in Diggle et al. (1994), Yao et al. (2005), Peng and Paul (2009) and Goldsmith et al. (2013).

For our analysis we consider $\log(\text{CD4 count})$ since the counts are skewed. We plot the data in Figure 9 where the x-axis is months since seroconversion (i.e., the time at which HIV becomes detectable). The overall trend seem to be decreasing, as can be confirmed by the estimated mean function plotted in Figure 9. The estimated variance and correlation functions are displayed in Figure 10. It is interesting to see that the minimal value of the estimated variance function occurs at month 0 since seroconversion. Finally we display in Figure 11 the predicted trajectory of $\log(\text{CD4 count})$ for 4 males and the corresponding pointwise confidence bands.

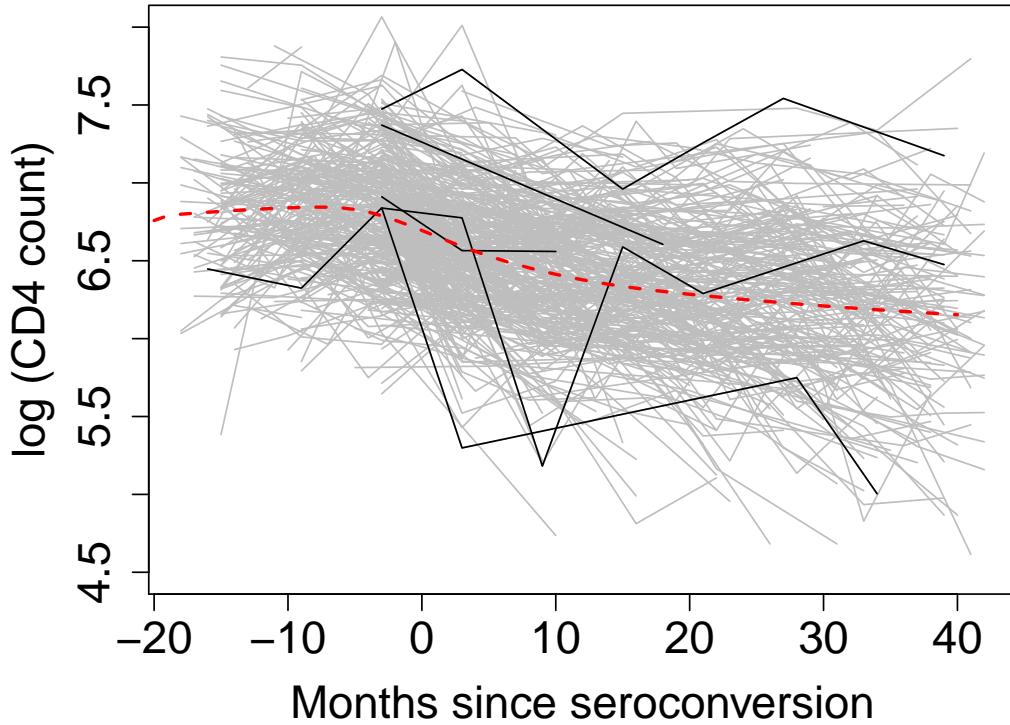


Figure 9: Observed $\log(\text{CD4 count})$ trajectories of 366 HIV-infected males. The estimated population mean is the black solid line.

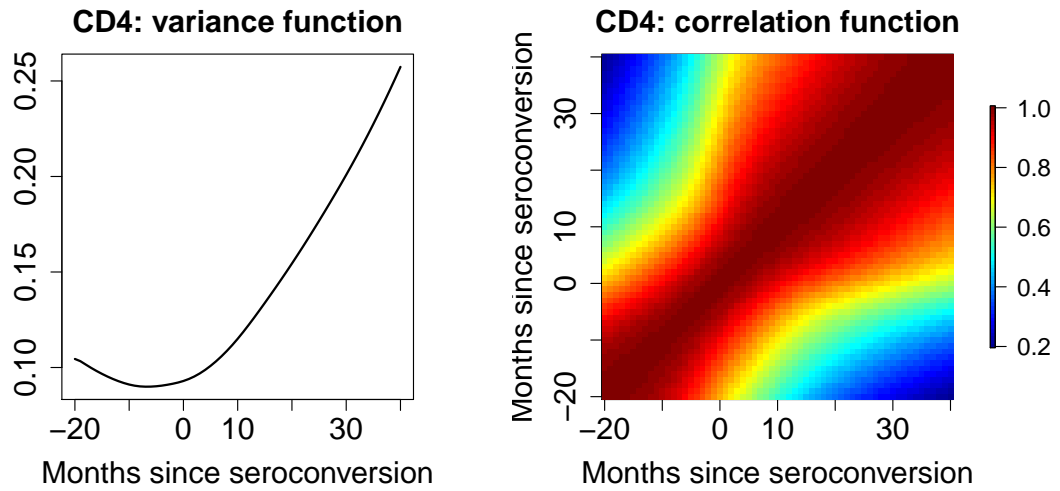


Figure 10: Estimated variance function (left panel) and correlation function (right panel) for the log (CD4 count).

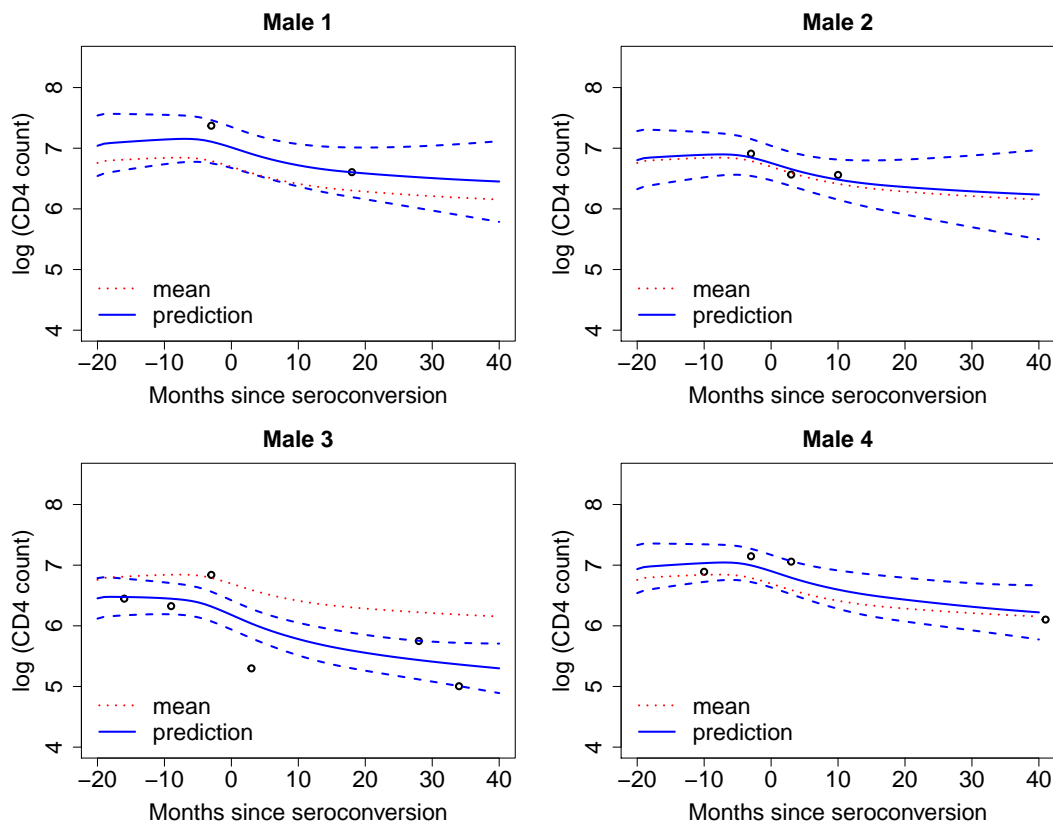


Figure 11: Predicted subject-specific trajectories of log (CD4 count) and associated 95% confidence bands for 4 males. The estimated population mean is the dotted line.

7 Discussion

Estimating and smoothing covariance operators is an old problem with many proposed solutions. Automatic and fast covariance smoothing is not fully developed and, in practice, one still does not have a method that is used consistently. The reason why the practical solution to the problem has been quite elusive is the lack of automatic covariance smoothing software. The novelty of our proposal is that it directly tackles this problem from the point of view of practicality. Here we proposed a method that we are already using extensively in practice and which is becoming increasingly popular among practitioners.

The ingredients of the proposed approach are not all new, but their combination leads to a complete product that can be used in practice. The fundamentally novel contributions that make everything practical are: 1) use a particular type of penalty that respects the covariance matrix format; 2) provide a very fast fitting algorithm for leave-one-subject-out cross validation; and 3) ensure the scalability of the approach by controlling the overall complexity of the algorithm.

To make methods transparent and reproducible, the method will be made publicly available soon in the *face* package and will be incorporated in the function *fpca.face* in the *refund* package later. The current *fpca.face* function (Xiao et al., 2014) deals with high-dimensional functional data observed on the same grid and has been used extensively by our collaborators. We have a long track-record of releasing functional data analysis software and the final form of the function will be part of the next release of *refund*.

Appendix

Proofs

Proof of Proposition 1: We will use the following Lemma (Isserlis, 1918) in our proof.

Lemma 2. If (x_1, \dots, x_{2n}) is a zero mean multivariate normal random vector, then

$$\mathbb{E}(x_1, \dots, x_{2n}) = \sum \prod \mathbb{E}(x_i x_j),$$

where $x_i x_j$ are all distinct pairs over x_1, x_2, \dots, x_{2n} .

First, we have

$$\begin{aligned} \text{Cov}(\hat{C}_{ijj'}, \hat{C}_{ikk'}) &= \mathbb{E}(\hat{C}_{ijj'} \hat{C}_{ikk'}) - \mathbb{E}(\hat{C}_{ijj'}) \mathbb{E}(\hat{C}_{ikk'}), \\ \mathbb{E}(\hat{C}_{ijj'}) &= \mathcal{C}(t_{ij}, t_{ij'}) + \delta_{jj'} \sigma_\epsilon^2, \\ \mathbb{E}(\hat{C}_{ikk'}) &= \mathcal{C}(t_{ik}, t_{ik'}) + \delta_{kk'} \sigma_\epsilon^2. \end{aligned}$$

Then,

$$\begin{aligned} &\mathbb{E}(\hat{C}_{ijj'}) \mathbb{E}(\hat{C}_{ikk'}) \\ &= \mathcal{C}(t_{ij}, t_{ij'}) \mathcal{C}(t_{ik}, t_{ik'}) + \mathcal{C}(t_{ij}, t_{ij'}) \delta_{kk'} \sigma_\epsilon^2 + \mathcal{C}(t_{ik}, t_{ik'}) \delta_{jj'} \sigma_\epsilon^2 + \delta_{jj'} \sigma_\epsilon^2 \delta_{kk'} \sigma_\epsilon^2. \end{aligned}$$

Next, we derive $\mathbb{E}(\hat{C}_{ijj'} \hat{C}_{ikk'})$ that

$$\begin{aligned} &\mathbb{E}(\hat{C}_{ijj'} \hat{C}_{ikk'}) \\ &= \mathbb{E}(y_{ij} y_{ij'} y_{ik} y_{ik'}) \\ &= \mathbb{E}\{(x_i(t_{ij}) + \epsilon_{ij})(x_i(t_{ij'}) + \epsilon_{ij'})(x_i(t_{ik}) + \epsilon_{ik})(x_i(t_{ik'}) + \epsilon_{ik'})\} \\ &= \mathbb{E}\{x_i(t_{ij}) x_i(t_{ij'}) x_i(t_{ik}) x_i(t_{ik'})\} + \mathbb{E}\{x_i(t_{ij}) x_i(t_{ij'}) x_i(t_{ik})\} \mathbb{E}(\epsilon_{ik'}) \\ &\quad + \mathbb{E}\{x_i(t_{ij}) x_i(t_{ij'}) x_i(t_{ik'})\} \mathbb{E}(\epsilon_{ik}) + \mathbb{E}\{x_i(t_{ij}) x_i(t_{ij'})\} \mathbb{E}(\epsilon_{ik} \epsilon_{ik'}) \\ &\quad + \mathbb{E}\{x_i(t_{ij}) x_i(t_{ik}) x_i(t_{ik'})\} \mathbb{E}(\epsilon_{ij'}) + \mathbb{E}\{x_i(t_{ij}) x_i(t_{ik})\} \mathbb{E}(\epsilon_{ij'} \epsilon_{ik'}) \\ &\quad + \mathbb{E}\{x_i(t_{ij}) x_i(t_{ik'})\} \mathbb{E}(\epsilon_{ij'} \epsilon_{ik}) + \mathbb{E}\{x_i(t_{ij})\} \mathbb{E}(\epsilon_{ij'} \epsilon_{ik} \epsilon_{ik'}) \\ &\quad + \mathbb{E}\{x_i(t_{ij'}) x_i(t_{ik}) x_i(t_{ik'})\} \mathbb{E}(\epsilon_{ij}) + \mathbb{E}\{x_i(t_{ij'}) x_i(t_{ik})\} \mathbb{E}(\epsilon_{ij} \epsilon_{ik'}) \\ &\quad + \mathbb{E}\{x_i(t_{ij'}) x_i(t_{ik'})\} \mathbb{E}(\epsilon_{ij} \epsilon_{ik}) + \mathbb{E}\{x_i(t_{ij'})\} \mathbb{E}(\epsilon_{ij} \epsilon_{ik} \epsilon_{ik'}) \\ &\quad + \mathbb{E}\{x_i(t_{ik}) x_i(t_{ik'})\} \mathbb{E}(\epsilon_{ij} \epsilon_{ij'}) + \mathbb{E}\{x_i(t_{ik})\} \mathbb{E}(\epsilon_{ij} \epsilon_{ij'} \epsilon_{ik'}) \\ &\quad + \mathbb{E}\{x_i(t_{ik'})\} \mathbb{E}(\epsilon_{ij} \epsilon_{ij'} \epsilon_{ik}) + \mathbb{E}(\epsilon_{ij} \epsilon_{ij'} \epsilon_{ik} \epsilon_{ik'}) \\ &= \mathbb{E}\{x_i(t_{ij}) x_i(t_{ij'}) x_i(t_{ik}) x_i(t_{ik'})\} + \mathbb{E}(\epsilon_{ij} \epsilon_{ij'} \epsilon_{ik} \epsilon_{ik'}) \\ &\quad + \mathcal{C}(t_{ij}, t_{ij'}) \delta_{kk'} \sigma_\epsilon^2 + \mathcal{C}(t_{ij}, t_{ik}) \delta_{j'k'} \sigma_\epsilon^2 + \mathcal{C}(t_{ij}, t_{ik'}) \delta_{jk} \sigma_\epsilon^2 \\ &\quad + \mathcal{C}(t_{ij'}, t_{ik}) \delta_{jk'} \sigma_\epsilon^2 + \mathcal{C}(t_{ij'}, t_{ik'}) \delta_{jk} \sigma_\epsilon^2 + \mathcal{C}(t_{ik}, t_{ik'}) \delta_{jj'} \sigma_\epsilon^2. \end{aligned}$$

Finally, with $\mathbb{E}(\hat{C}_{ijj'})\mathbb{E}(\hat{C}_{ikk'})$ and $\mathbb{E}(\hat{C}_{ijj'}\hat{C}_{ikk'})$, we have

$$\begin{aligned}
\text{Cov}(\hat{C}_{ijj'}, \hat{C}_{ikk'}) &= \mathcal{C}(t_{ij}, t_{ij'})\mathcal{C}(t_{ik}, t_{ik'}) + \mathcal{C}(t_{ij}, t_{ik})\mathcal{C}(t_{ij'}, t_{ik'}) + \mathcal{C}(t_{ij}, t_{ik'})\mathcal{C}(t_{ij'}, t_{ik}) \\
&+ \delta_{jj'}\sigma_\epsilon^2\delta_{kk'}\sigma_\epsilon^2 + \delta_{jk}\sigma_\epsilon^2\delta_{j'k'}\sigma_\epsilon^2 + \delta_{jk'}\sigma_\epsilon^2\delta_{j'k}\sigma_\epsilon^2 \\
&+ \mathcal{C}(t_{ij}, t_{ij'})\delta_{kk'}\sigma_\epsilon^2 + \mathcal{C}(t_{ij}, t_{ik})\delta_{j'k'}\sigma_\epsilon^2 + \mathcal{C}(t_{ij}, t_{ik'})\delta_{j'k}\sigma_\epsilon^2 \\
&+ \mathcal{C}(t_{ij'}, t_{ik})\delta_{jk'}\sigma_\epsilon^2 + \mathcal{C}(t_{ij'}, t_{ik'})\delta_{jk}\sigma_\epsilon^2 + \mathcal{C}(t_{ik}, t_{ik'})\delta_{jj'}\sigma_\epsilon^2 \\
&- \mathcal{C}(t_{ij}, t_{ij'})\mathcal{C}(t_{ik}, t_{ik'}) - \mathcal{C}(t_{ij}, t_{ij'})\delta_{kk'}\sigma_\epsilon^2 - \mathcal{C}(t_{ik}, t_{ik'})\delta_{jj'}\sigma_\epsilon^2 - \delta_{jj'}\sigma_\epsilon^2\delta_{kk'}\sigma_\epsilon^2 \\
&= \mathcal{C}(t_{ij}, t_{ik})\mathcal{C}(t_{ij'}, t_{ik'}) + \mathcal{C}(t_{ij}, t_{ik'})\mathcal{C}(t_{ij'}, t_{ik}) + \delta_{jk}\delta_{j'k'}\sigma_\epsilon^4 + \delta_{jk'}\delta_{j'k}\sigma_\epsilon^4 \\
&+ \mathcal{C}(t_{ij}, t_{ik})\delta_{j'k'}\sigma_\epsilon^2 + \mathcal{C}(t_{ij}, t_{ik'})\delta_{j'k}\sigma_\epsilon^2 + \mathcal{C}(t_{ij'}, t_{ik})\delta_{jk'}\sigma_\epsilon^2 + \mathcal{C}(t_{ij'}, t_{ik'})\delta_{jk}\sigma_\epsilon^2.
\end{aligned}$$

which proves the proposition.

Proof of Proposition 2: We will use the following Lemma (page 241, Seber 2007).

Lemma 3. *Let \mathbf{A} , \mathbf{B} and \mathbf{C} and \mathbf{D} be compatible matrices. Then*

$$\text{tr}(\mathbf{ABCD}) = (\text{vec } \mathbf{D})^T (\mathbf{A} \otimes \mathbf{C}^T) \text{vec } \mathbf{B}^T.$$

We first write iGCV as a sum

$$\text{iGCV} = \mathcal{I} + 2 \sum_{i=1}^n (\mathcal{II}_i - 2\mathcal{III}_i + \mathcal{IV}_i), \quad (8)$$

where

$$\begin{aligned}
\mathcal{I} &= \|\hat{\mathbf{C}} - \mathbf{S}\hat{\mathbf{C}}\|^2, \\
\mathcal{II}_i &= \hat{\mathbf{C}}_i^T \mathbf{S}_{ii} \hat{\mathbf{C}}_i, \\
\mathcal{III}_i &= (\mathbf{S}_i \hat{\mathbf{C}})^T \mathbf{S}_{ii} \hat{\mathbf{C}}_i, \\
\mathcal{IV}_i &= (\mathbf{S}_i \hat{\mathbf{C}})^T \mathbf{S}_{ii} (\mathbf{S}_i \hat{\mathbf{C}}).
\end{aligned}$$

Note that we have the following equalities that will be used later:

$$\begin{aligned}
\mathbf{S} &= (\mathbf{XA})[\mathbf{I} + \lambda \text{diag}(\mathbf{s})]^{-1} (\mathbf{XA})^T \mathbf{W} \\
&= \mathbf{F} \tilde{\mathbf{D}} \tilde{\mathbf{F}},
\end{aligned}$$

$$\begin{aligned}
\mathbf{S}_i &= (\mathbf{X}_i \mathbf{A})[\mathbf{I} + \lambda \text{diag}(\mathbf{s})]^{-1} (\mathbf{XA})^T \mathbf{W} \\
&= \mathbf{F}_i \tilde{\mathbf{D}} \tilde{\mathbf{F}},
\end{aligned}$$

and

$$\begin{aligned}\mathbf{S}_{ii} &= (\mathbf{X}_i \mathbf{A})[\mathbf{I} + \lambda \text{diag}(\mathbf{s})]^{-1} (\mathbf{X}_i \mathbf{A})^T \mathbf{W}_i \\ &= \mathbf{F}_i \tilde{\mathbf{D}} \mathbf{F}_i^T \mathbf{W}_i.\end{aligned}$$

We first compute \mathcal{I} . We have

$$\begin{aligned}\mathcal{I} &= \|\hat{\mathbf{C}} - \mathbf{S}\hat{\mathbf{C}}\|^2 \\ &= \|\hat{\mathbf{C}} - \mathbf{F}\tilde{\mathbf{D}}\tilde{\mathbf{f}}\|^2 \\ &= \|\hat{\mathbf{C}}\|^2 - 2\tilde{\mathbf{f}}^T \tilde{\mathbf{D}}\tilde{\mathbf{f}} + \tilde{\mathbf{f}}^T \tilde{\mathbf{D}} \mathbf{F}^T \mathbf{F} \tilde{\mathbf{D}}\tilde{\mathbf{f}}.\end{aligned}$$

Thus,

$$\mathcal{I} = \|\hat{\mathbf{C}}\|^2 - 2\tilde{\mathbf{d}}^T (\tilde{\mathbf{f}} \odot \mathbf{f}) + (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}})^T (\mathbf{F}^T \mathbf{F}) (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}}). \quad (9)$$

Second, we compute \mathcal{II}_i . We have

$$\mathcal{II}_i = \hat{\mathbf{C}}_i^T \mathbf{S}_{ii} \hat{\mathbf{C}}_i = \mathbf{f}_i^T \tilde{\mathbf{D}} \mathbf{F}_i^T \mathbf{W}_i \hat{\mathbf{C}}_i = \mathbf{f}_i^T \tilde{\mathbf{D}} \mathbf{J}_i = \tilde{\mathbf{d}}^T (\mathbf{J}_i \odot \mathbf{f}_i). \quad (10)$$

Third, we compute \mathcal{III}_i . Note that $\mathbf{S}_i \hat{\mathbf{C}} = \mathbf{F}_i \tilde{\mathbf{D}} \tilde{\mathbf{f}}$ and hence

$$\begin{aligned}\mathcal{III}_i &= (\mathbf{S}_i \hat{\mathbf{C}})^T \mathbf{S}_{ii} \hat{\mathbf{C}}_i \\ &= \tilde{\mathbf{f}}^T \tilde{\mathbf{D}} \mathbf{F}_i^T \mathbf{F}_i \tilde{\mathbf{D}} \mathbf{J}_i \\ &= \text{tr}(\mathbf{J}_i \tilde{\mathbf{f}}^T \tilde{\mathbf{D}} \mathbf{F}_i^T \mathbf{F}_i \tilde{\mathbf{D}}) \\ &= \tilde{\mathbf{d}}^T \{(\mathbf{J}_i \tilde{\mathbf{f}}^T) \odot (\mathbf{F}_i^T \mathbf{F}_i)\} \tilde{\mathbf{d}}.\end{aligned}$$

Thus we have

$$\mathcal{III}_i = \tilde{\mathbf{d}}^T \{(\mathbf{J}_i \tilde{\mathbf{f}}^T) \odot (\mathbf{F}_i^T \mathbf{F}_i)\} \tilde{\mathbf{d}}. \quad (11)$$

Fourth, we compute \mathcal{IV}_i . We derive that

$$\begin{aligned}\mathcal{IV}_i &= (\mathbf{S}_i \hat{\mathbf{C}})^T \mathbf{S}_{ii} (\mathbf{S}_i \hat{\mathbf{C}}) \\ &= \tilde{\mathbf{f}}^T \tilde{\mathbf{D}} \mathbf{F}_i^T \mathbf{F}_i \tilde{\mathbf{D}} \mathbf{F}_i^T \mathbf{W}_i \mathbf{F}_i \tilde{\mathbf{D}} \tilde{\mathbf{f}} \\ &= \text{tr}(\tilde{\mathbf{f}}^T \tilde{\mathbf{D}} \mathbf{F}_i^T \mathbf{F}_i \tilde{\mathbf{D}} \mathbf{F}_i^T \mathbf{W}_i \mathbf{F}_i \tilde{\mathbf{D}} \tilde{\mathbf{f}}) \\ &= \text{tr}(\mathbf{F}_i^T \mathbf{F}_i \tilde{\mathbf{D}} \mathbf{L}_i \tilde{\mathbf{D}} \tilde{\mathbf{f}} \tilde{\mathbf{f}}^T \tilde{\mathbf{D}}) \\ &= \text{tr}(\tilde{\mathbf{D}} \mathbf{L}_i (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}}) (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}})^T \mathbf{F}_i^T \mathbf{F}_i) \\ &= \tilde{\mathbf{d}}^T \left[\left\{ \mathbf{L}_i (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}}) \right\} \odot \left\{ (\mathbf{F}_i^T \mathbf{F}_i) (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}}) \right\} \right].\end{aligned}$$

Hence we obtain

$$\mathcal{IV}_i = \tilde{\mathbf{d}}^T \{ \mathbf{L}_i \odot (\mathbf{F}_i^T \mathbf{F}_i) \} \left\{ (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}}) \otimes (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}}) \right\}. \quad (12)$$

Now with (8), (9), (10), (11) and (12), we obtain that

$$\begin{aligned} \text{iGCV} &= \|\hat{\mathbf{C}}\|^2 - 2\tilde{\mathbf{d}}^T (\tilde{\mathbf{f}} \odot \mathbf{f}) + (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}})^T (\mathbf{F}^T \mathbf{F}) (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}}) + 2 \sum_{i=1}^n \tilde{\mathbf{d}}^T (\mathbf{J}_i \odot \mathbf{f}_i) \\ &\quad + 2 \sum_{i=1}^n \left[-2\tilde{\mathbf{d}}^T \left\{ (\mathbf{J}_i \tilde{\mathbf{f}}^T) \odot (\mathbf{F}_i^T \mathbf{F}_i) \right\} \tilde{\mathbf{d}} + \tilde{\mathbf{d}}^T \{ \mathbf{L}_i \odot (\mathbf{F}_i^T \mathbf{F}_i) \} \left\{ (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}}) \otimes (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}}) \right\} \right] \\ &= \|\hat{\mathbf{C}}\|^2 - 2\tilde{\mathbf{d}}^T (\tilde{\mathbf{f}} \odot \mathbf{f}) + (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}})^T (\mathbf{F}^T \mathbf{F}) (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}}) + 2\tilde{\mathbf{d}}^T \left\{ \sum_{i=1}^n \mathbf{J}_i \odot \mathbf{f}_i \right\} \\ &\quad - 4\tilde{\mathbf{d}}^T \left\{ \sum_{i=1}^n (\mathbf{J}_i \tilde{\mathbf{f}}^T) \odot (\mathbf{F}_i^T \mathbf{F}_i) \right\} \tilde{\mathbf{d}} + 2\tilde{\mathbf{d}}^T \left\{ \sum_{i=1}^n \mathbf{L}_i \odot (\mathbf{F}_i^T \mathbf{F}_i) \right\} \left\{ (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}}) \otimes (\tilde{\mathbf{f}} \odot \tilde{\mathbf{d}}) \right\}, \end{aligned}$$

which proves the proposition.

References

- Besse, P., H. Cardot, and F. Ferraty (1997). Simultaneous nonparametric regressions of unbalanced longitudinal data. *Comput. Statist. Data Anal.* *24*, 255–270.
- Besse, P. and J. O. Ramsay (1986). Principal components analysis of sampled functions. *Psychometrika* *51*, 285–311.
- Diggle, P., P. Heagerty, K.-Y. Liang, and S. Zeger (1994). *Analysis of longitudinal data*. Oxford, U.K.: Oxford University Press.
- Eilers, P. and B. Marx (1996). Flexible smoothing with B-splines and penalties (with Discussion). *Statist. Sci.* *11*, 89–121.
- Eilers, P. and B. Marx (2003). Multivariate calibration with temperature interaction using two-dimensional penalized signal regression. *Chemometrics and Intelligent Laboratory Systems* *66*, 159–174.
- Fan, J. and I. Gijbels (1996). *Local polynomial modelling and its applications*. London: Chapman&Hall/CRC.
- Goldsmith, J., J. Bobb, C. Crainiceanu, B. Caffo, and D. Reich (2010). Penalized functional regression. *J. Comput. Graph. Statist.* *20*, 830–851.
- Goldsmith, J., S. Greven, and C. Crainiceanu (2013). Corrected confidence bands for functional data using principal components. *Biometrics* *69*(1), 41–51.
- Huang, L., F. Scheipl, J. Goldsmith, J. Gellar, J. Harezlak, M. Mclean, B. Swihart, L. Xiao, C. Crainiceanu, P. Reiss, Y. Chen, S. Greven, L. Huo, M. Kundu, and J. Wrobel (2015). R package *mgcv*: Methodology for regretssion with functional data (version 0.1-13). URL: <https://cran.r-project.org/web/packages/refund/index.html>.

- Isserlis, L. (1918, November). On a formula for the product-moment coefficient of any order of a normal frequency distribution in any number of variables. *Biometrika* 12(1-2), 134–139.
- James, G., T. Hastie, and C. Sugar (2000). Principal component models for sparse functional data. *Biometrika* 87, 587–602.
- Kaslow, R. A., D. G. Ostrow, R. Detels, J. P. Phair, B. F. Polk, and C. R. Rinaldo (1987). The multicenter aids cohort study: Rationale, organization, and selected characteristics of the participants. *American Journal of Epidemiology* 126(2), 310–318.
- Kneip, A. (1994). Nonparametric estimation of common regressors for similar curve data. *Ann. Statist.* 22, 1386–1427.
- Peng, J. and D. Paul (2009). A geometric approach to maximum likelihood estimation of functional principal components from sparse longitudinal data. *J. Comput. Graph. Stat.* 18, 995–1015.
- Ramsay, J. and C. J. Dalzell (1991). Some tools for functional data analysis (with Discussion). *J. R. Statist. Soc. B* 53, 539–572.
- Reiss, P., L. Huang, and M. Mennes (2010). Fast function-on-scalar regression with penalized basis expansions. *Int. J. Biostat.* 6, 28.
- Seber, G. (2007). *A Matrix Handbook for Statisticians*. New Jersey: Wiley-Interscience.
- Staniswalis, J. and J. Lee (1998). Nonparametric regression analysis of longitudinal data. *J. Amer. Statist. Assoc.* 93, 1403–1418.
- Wood, S. (2003). Thin plate regression splines. *J. R. Statist. Soc. B* 65, 95–114.
- Wood, S. (2013). R package *mgcv*: Mixed GAM computation vehicle with GCV/AIC/REML, smoothest estimation (version 1.7-24). URL:<http://cran.r-project.org/web/packages/mgcv/index.html>.

- Xiao, L., L. Huang, J. Schrack, L. Ferrucci, V. Zipunnikov, and C. Crainiceanu (2015). Quantifying the life-time circadian rhythm of physical activity: a covariate-dependent functional approach. *Biostatistics* 16, 352–367.
- Xiao, L., Y. Li, and D. Ruppert (2013). Fast bivariate P -splines: the sandwich smoother. *J. R. Statist. Soc. B* 75, 577–599.
- Xiao, L., D. Ruppert, V. Zipunnikov, and C. Crainiceanu (2014). Fast covariance function estimation for high-dimensional functional data. *Stat. Comput.* 26, 409–421.
- Xu, G. and J. Huang (2012). Asymptotic optimality and efficient computation of the leave-subject-out cross-validation. *Ann. Statist.* 40, 3003–3030.
- Yao, F., H. Müller, A. Clifford, S. Dueker, J. Follett, Y. Lin, B. Buchholz, and J. Vogel (2003). Shrinkage estimation for functional principal component scores with application to the population kinetics of plasma folate. *Biometrics* 20, 852–873.
- Yao, F., H. Müller, and J. Wang (2005). Functional data analysis for sparse longitudinal data. *J. Amer. Statist. Assoc.* 100, 577–590.